LIFE User's Guide

The Planck/LFI People

This manual is for LIFE, version 3 (Lama 3.1.0, 2008-09-05 15:36:10).

Copyright © 2004-2008 The Planck/LFI People.

First version created by Maurizio Tomasi on 2006/06/22.

Table of Contents

1 Overview 1
- 1.1 RaNA 1
- 1.2 LAMA 1
- 1.3 OCA 1
- 1.4 Conventions used in this manual 1

2 Tutorial 3
- 2.1 Getting Help 3
- 2.2 A Lama View Session 3

3 RaNA 8

4 LAMA 9
- 4.1 The Lama Top Window 9
  - 4.1.1 Start Session 9
  - 4.1.2 View Data 9
  - 4.1.3 Analyse Data 9
  - 4.1.4 Report 9
  - 4.1.5 Close Session 9
  - 4.1.6 Additional Buttons 9
- 4.2 The Lama View Window 10
  - 4.2.1 Node List 10
    - 4.2.1.1 Add new test 11
    - 4.2.1.2 Unload test 11
    - 4.2.1.3 Add Data to New Plot 11
    - 4.2.1.4 Add data to an Existing Plot 12
    - 4.2.1.5 The Default Selection 12
    - 4.2.1.6 Right Mouse Button Menu 12
  - 4.2.2 Window List 13
- 4.3 The Lama Plot Window 13
  - 4.3.1 The plot 13
    - 4.3.1.1 Zoom data 14
    - 4.3.1.2 Navigating into the data 14
    - 4.3.1.3 Select data 14
    - 4.3.1.4 Pan plot 15
    - 4.3.1.5 Reset plot 15
    - 4.3.1.6 The Plot Legend 15
  - 4.3.2 Operations 15
  - 4.3.3 Scientific Selections 16
  - 4.3.4 Radiometric Operations 17
    - 4.3.4.1 Swap sky and reference 18
    - 4.3.4.2 Show gaps in scientific data 18

4.3.5 Housekeeping information . . . 18
4.3.6 Manual operations . . . 19
4.4 The Lama Cross-correlation Window . . . 20
4.4.1 Choosing Data . . . 20
4.4.2 The plot . . . 20
4.4.2.1 Navigating into the data . . . 21
4.4.2.2 Save data . . . 21
4.4.3 Time Span . . . 21
4.4.4 Operations . . . 21
4.4.4.1 Reset . . . 21
4.4.4.2 Swap Axes . . . 21
4.4.4.3 Show points . . . 22
4.4.4.4 Show error bars . . . 22
4.4.4.5 Bin data . . . 22
4.4.4.6 Show fit . . . 22
4.4.5 Data Information Window . . . 22

5 OCA . . . 24

6 IDL Command Reference . . . 25

6.1 The Lama Instance . . . 25
6.2 Opening and closing Lama View . . . 25
6.3 Loading/unloading tests . . . 26
6.4 Feed horns . . . 27
6.5 Accessing data . . . 28
6.5.1 Scientific data . . . 28
6.5.1.1 General functions . . . 28
6.5.1.2 Accessing binned data . . . 29
6.5.1.3 Accessing full scientific data . . . 30
6.5.1.4 Statistics . . . 32
6.5.1.5 Photometric calibration . . . 34
6.5.1.6 Miscellanea . . . 34
6.5.2 Housekeeping data . . . 36
6.5.2.1 Housekeeping Statistics . . . 37
6.5.2.2 Miscellanea . . . 37
6.5.3 Browsing the node tree . . . 38
6.5.3.1 Node paths . . . 39
6.5.3.2 The current node . . . 42
6.5.3.3 Retrieving scientific/housekeeping data . . . 43
6.5.3.4 Unloading data from memory . . . 43
6.5.3.5 Finding nodes in the tree . . . 44
6.5.4 Using measure units . . . 45
6.5.4.1 RaNA . . . 45
6.5.4.2 Lama . . . 45
6.6 Triggers . . . 46
6.7 Log messages . . . 48
6.8 DAE calibration . . . 50

Appendix A Regular expressions 52
A.1 Backreferences 54

Concept index 56

Function index 58

1 Overview

This is the LFI User's Guide, a document which illustrates the features of LIFE 3. It is in a very rough format, but it provides some useful information about LIFE from the point of view of an user.

There is no need to be a programmer to read this manual. However, knowing the IDL language will help reading those sections which describe the interface between LIFE and IDL.

LIFE is a package of a number of different tools. They are RaNA (used to test the LFI RCAs), LAMA (used to test the two LFI RAAs) and OCA (introduced with LAMA 2.0, it is a tool written by Michele Maris to test the REBA).

1.1 RaNA

RaNA, the Radiometer aNAlyser, is a tool used during the LFI QM/FM RCA test campaigns. It opens FITS files produced by Rachel and let the user to calibrate the chains and perform a number of operations on the scientific/housekeeping data.

1.2 LAMA

LAMA (the LFI Array Measurement Analyser) is a tool for the LFI QM/FM RAA test campaigns. It can read the FITS files created by TM2TOI and perform a number of analysis similar to RaNA.

In order to simplify the analisys for the 44 FM RCAs, the analysis modules now have a command-line interface which eases the use of scripts to perform automatic procedures.

1.3 OCA

OCA is a tool written by Michele Maris (maris@oats.inaf.it) to test the REBA.

1.4 Conventions used in this manual

This reference lists the IDL procedures and functions defined by LIFE and accessible through the IDL command line. Here is a list of the conventions we used in this manual:

- IDL procedures are marked with "Procedure". They never return a value and are called by following their name with a command and a list of arguments. An example is [lama_instance_close], page 25.
- IDL functions are marked as "Function". Their return value is described in the text following the function definition. When calling a function, you must provide the arguments enclosed into rounded brackets. Brackets should be used even when the function has no arguments. An example is [lama_instance_is_created], page 25.
- When providing an example, we use the ⊣ symbol to indicate what is the output of the commands, e.g.:

```
print, 'Hello, world!'
⊣ Hello, world!
```

- Every procedure and function described in this manual is listed in the [Function index], page 58 (if it is not, it is a bug!).

You can download a PDF version of this manual at
http://belzebu.lambrate.inaf.it/lifedocs/life/manual/life.pdf.

2 Tutorial

2.1 Getting Help

As well as reading this manual, it is useful to know how to access the IDL help files for each function. Use the show_help command

```
show_help,'lama_tune_phase_switch_currents',/helpinput,/helpoutput
```

This function will generate a help window containing information about the function you specify.

2.2 A Lama View Session

First of all a test is needed. For this example download TUN_0015.tgz from Max and decompress it in a directory.

From command line execute:

```
idl run-life.pro
```

IDL compiles LIFE and after this, if there are no errors, the main LIFE window appears. Click on LAMA icon and Lama Top appears (Figure 2.1).

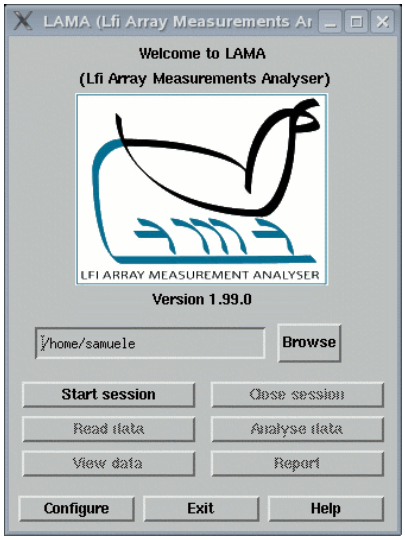

Figure 2.1: Lama Top main window

The first task is to open a session. All the operations done during the session are stored in a folder. Click on the button labeled **Browse** to change the location of the directory. Then click the button labeled **Start Session** to start the session. Lama creates the new directory and all subdirectories needed for saving operations.

Click the button labeled **View Data** to open Lama View. The window is empty. For loading the test click the button labeled **Add New Test** and select the directory in which are saved the TUN_0015 test. Click open. Now the test is opened and Lama View appears like in Figure 2.2.

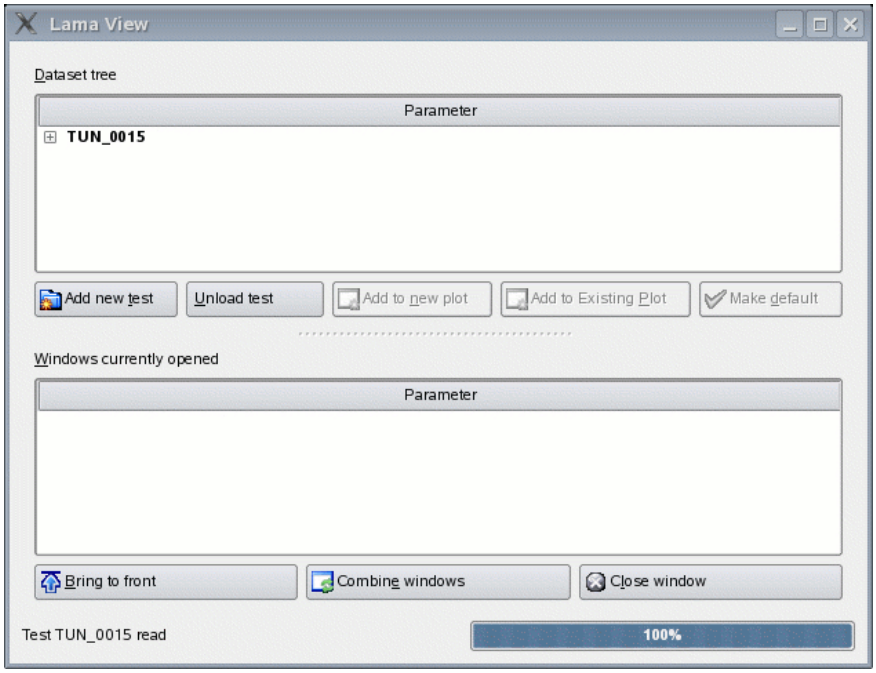

Figure 2.2: Lama View after the loading of the TUN_0015 test

Click on the + symbol to expand the tree. In this example we plot the feed horn 27. Select this feed horn and click the button labeled **Add to New Plot** as in Figure 2.3 to open the new window. Select in Node List with **ctrl** pressed other nodes to open more than one feed horn or housekeeping data.

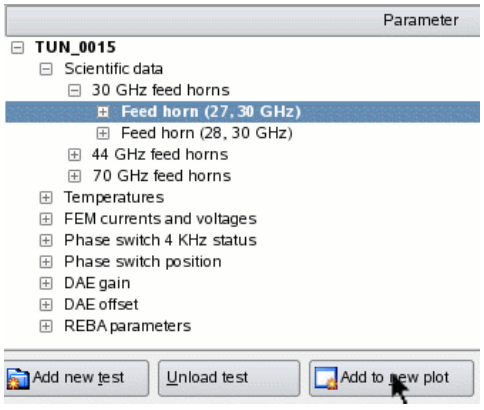

Figure 2.3: How to open new Lama Plot window

The new window appears like in Figure 2.4. We can see on the left the plot with the data opened, on the top left the possible plot operation, on the right one tab with different possibilities and the legend of the plot.

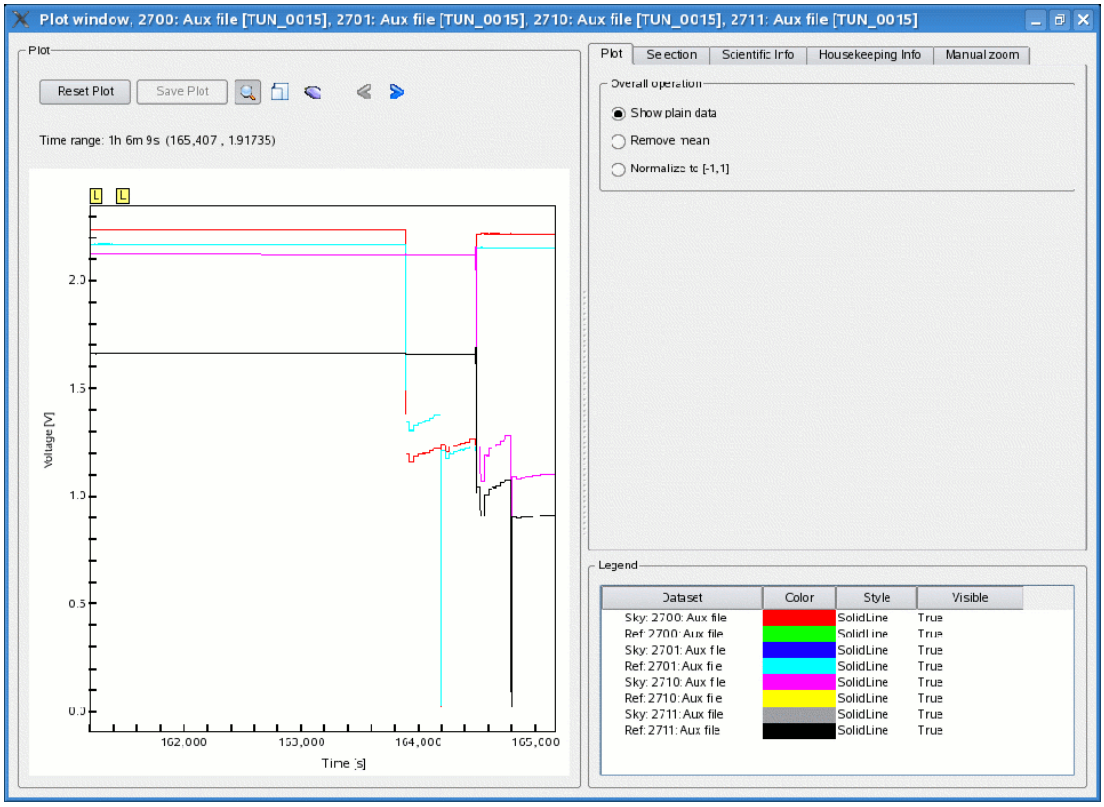

Figure 2.4: Lama Plot window

In the top left of the Lama Plot window there are six buttons. The buttons with the icons change the mouse interaction with the plot: the first is for zooming, the second is for selecting and the third is for pan. Press left mouse button on the plot and move it, a blue rectangle appears in the plot. Now release the button and the plot is zoomed in. Do the same pressing right mouse button for zooming out.
The two arrows let to navigate into the different zooms.
Now press the icon with the two sheet. Pressing the left mouse button on the plot and moving it, a blue rectangle shows the selection area (See [ScientificSelection], page 16, for details).
Now press the icon with the mouse. Pressing the left mouse button on the plot and moving it moves the entire plot.
Press the button labeled **Reset** to undo all the operations performed on the data and reset the zoom list.

On the right of the Lama Plot windows there is a tab with five pages.

- In the first page there are three radio buttons for seeing real data, data normalized in the range between -1 and +1, and data without theirs mean.
- The second page is for selection operation and works only with radiometric data. The radiometric data loaded and plotted by Lama View are AUX files, in which data are

binned at 1 Hz. The selection page let to load complete (or with a different bin value) data. See [ScientificSelection], page 16, for more information.

- The third and the fourth pages shows statistical information about plotted data. If the data are zoomed the statistics are referred to those data.
- The fifth page has a manual zoom and a button to select statistical range for IDL functions.

On the right bottom of the Lama Plot window there is the legend, in which are listed the data stream plotted with the name and the color. Select one or more lines and right click. The menu let to change style, color and hide (or show) the selected data stream.
Now select the first line (sky 2700) and the last line (ref 2711) and right click the mouse. Select Cross correlation from the menu. This opens a new window: the Crosscorrelation plot (Figure 2.5). This plot window shows data correlated. The main difference between this window and the Lama Plot window is in the first page of the tab in which it's possible to invert axes, to bin data, to show error bars. For more information See Section 4.4 [Cross], page 20.

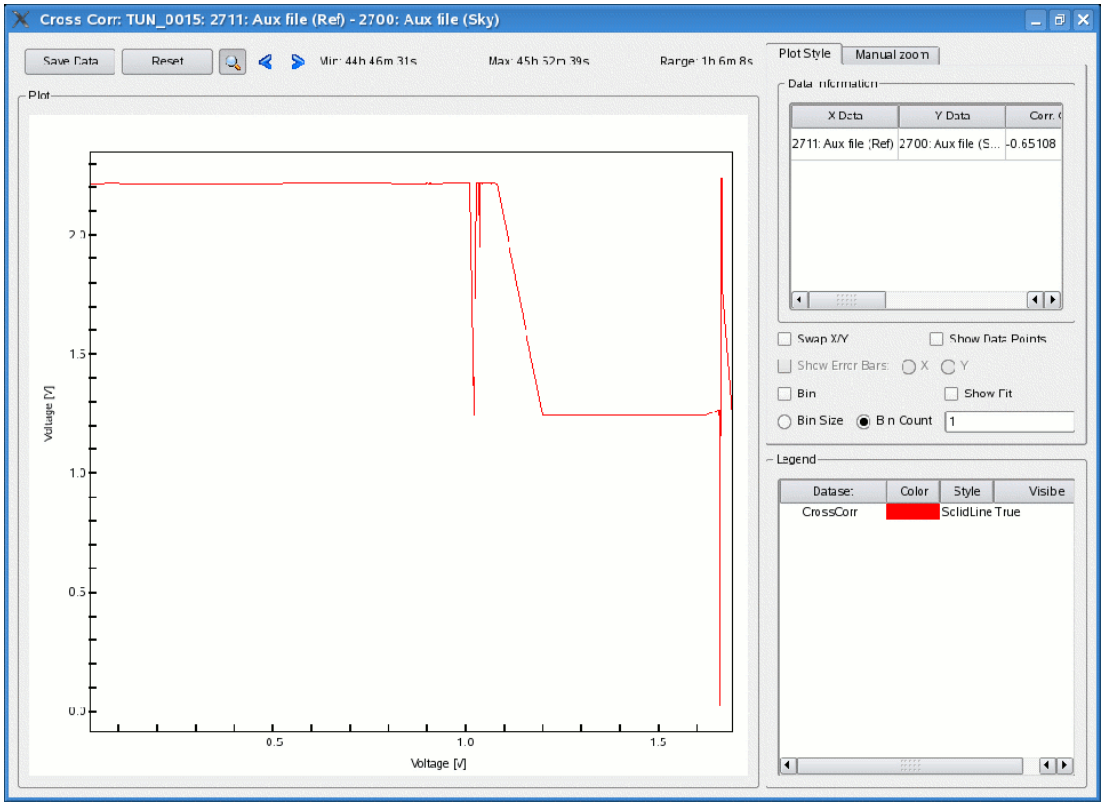

Figure 2.5: Lama Crosscorrelation plot window

Return to Lama View. Select the feed horn 28 and press the button labeled **Add to Existing**. In the menu select the voice Lama Plot 2700 window. Now in the bottom of the window there is a list. Select the voice Lama Plot 2700 window and press the button

labeled **Bring to Front**. The Lama Plot window shows data of both feed horns.
Return to Lama View. Select feed horn 24 and press the button labeled **Add to new plot**. Return to Lama View again. In the Window List select both windows and press the button labeled **Combine Windows**. Now there is only one window which shows data contained in both previous windows.

At the end of the work session close Lama View (this will close all the Lama Plot and Lama Cross opened windows) and return to Lama Top. Before exit the program close the session pressing the button labeled **Close Session** in Lama Top. Now exit the program.

For more details about Lama View see See Section 4.2 [View], page 10.

3 RaNA

4 LAMA

This chapter provides an overview of the LAMA (Lfi Array Measurement Analyser) software. It is divided into two main parts: the lama top section describes the functionality of the lama main window, the lama view section and the following describe how to access to the data and plot them.

The purpose of LAMA is to load and analyse data aquired during the LFI RAA FM test campaign. It provides a graphical user interface to load the data saved during the tests, plot scientific and housekeeping values, and provides a number of analysis tools to estimate the tuning parameters and the performances of the instrument.

4.1 The Lama Top Window

The main window of Lama is the Lama Top window (see image). Through this window the user can use Lama utilities: the Lama View software, the analyisis modules and the report.

The next sections describes how the buttons of the Lama Top window work.

4.1.1 Start Session

The first operation when the user open Lama is to start a session. A session is a folder where all the operations made during the work are stored. The predefined folder is the home folder but the user can change it by pressing the brwose button and choosing another folder. If the user doesn't open a session it's impossible to begin working. When a session is started Lama open the log messages window in which are stored all the software messages during the session.

4.1.2 View Data

The View Data button open the Lama view window, that is the software for plotting data (See Section 4.2 [The Lama View Window], page 10, for details).

4.1.3 Analyse Data

4.1.4 Report

4.1.5 Close Session

Before exit the program it's necessary to close the session in order to let LIFE to use others software. Pressing close session button the session is closed.

4.1.6 Additional Buttons

At bottom of the Lama Top window there are three additional button.

- The Configure button let the user to choose the LaTex software (for reporting operations), the DVI Reader (for reading the reports) and the internet browser.
- The Exit button close Lama. It doesn't work if one session is still opened.
- The Help button.

4.2 The Lama View Window

Lama view lets the user to plot data of the LFI instrument and the housekeeping data. The main window of Lama view shows two views: the node list and the windows list (Figure 4.1). The first view is a tree of nodes which represents scientific and housekeeping data, the second view is the list of the plot windows opened.

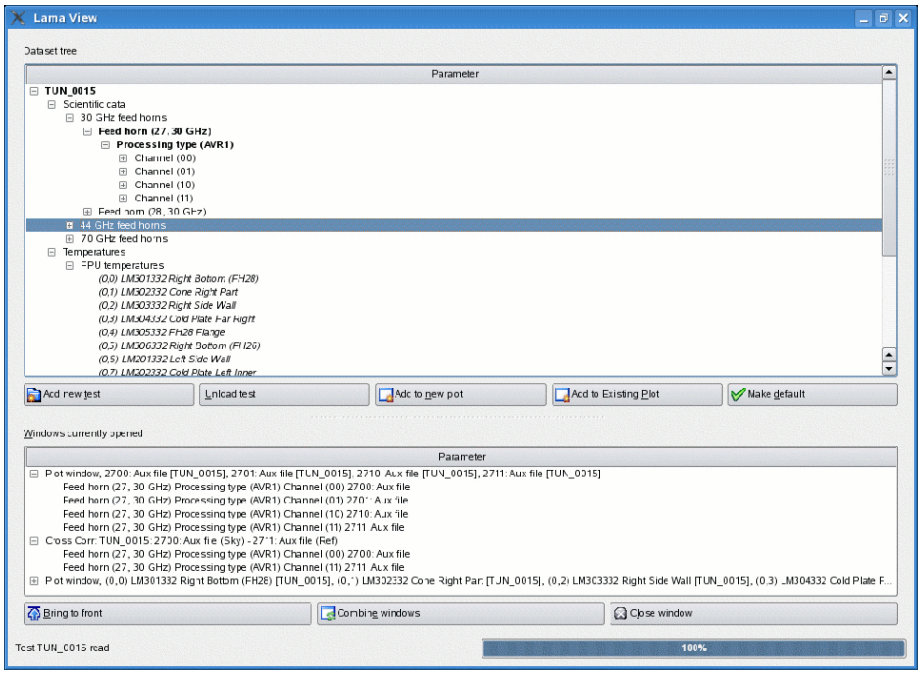
Lama View
Dataset tree
Parameter
TUN_0015
Scientific data
30 GHz feed horns
Feed horn (27, 30 GHz)
Processing type (AVR1)
Channel (00)
Channel (01)
Channel (10)
Channel (11)
44 GHz feed horns
70 GHz feed horns
Temperatures
Add new test
Unload test
Add to new plot
Add to Existing Plot
Make default
Windows currently opened
Parameter
Feed horn (27, 30 GHz) Processing type (AVR1) Channel (00) 2700: Aux file
Feed horn (27, 30 GHz) Processing type (AVR1) Channel (01) 2701: Aux file
Feed horn (27, 30 GHz) Processing type (AVR1) Channel (10) 2710: Aux file
Feed horn (27, 30 GHz) Processing type (AVR1) Channel (11) 2711 Aux file
Feed horn (27, 30 GHz) Processing type (AVR1) Channel (00) 2700: Aux file
Feed horn (27, 30 GHz) Processing type (AVR1) Channel (11) 2711 Aux file
Bring to front
Combine windows
Close window
Test TUN_0015 read
100%

Figure 4.1: Lama View main window

4.2.1 Node List

The node list shows a tree in which each node represents a scientific node or an housekeeping node (Figure 4.2). The root of the tree is the test of the desired data.
The scientific nodes are ordered by frequency (30 GHz, 44 GHz, 70 GHz), horn (from 18 to 28) and channel (4 for each horn). The housekeeping nodes are divided in groups: temperatures, voltage and currents of the FEM, DAE and REBA parameters.
Each node is selectable and must be selected in order to perform all the operations that are shown in the next sections.

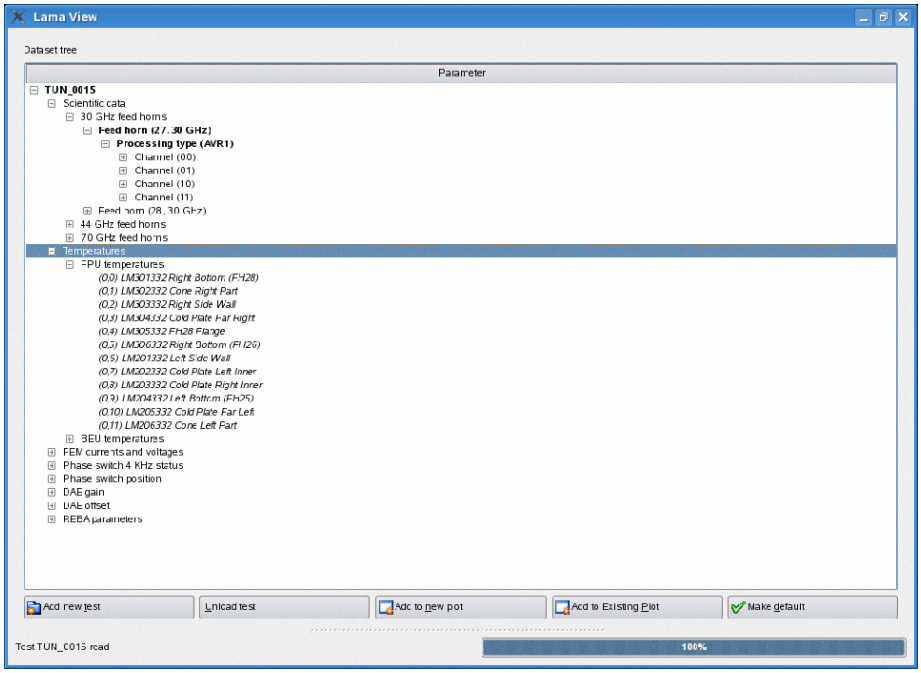

Figure 4.2: Node List in Lama View main window

4.2.1.1 Add new test

When Lama View is opened the Node List is void. In order to show some data is necessary that the user chosees a dataset. Pressing the button labeled **Add new test** (Figure 4.3) the software open a window where it's possible to choose the folder in which the desired data are stored. After this operation Lama View shows the new root of the dataset.

4.2.1.2 Unload test

When a dataset is no more usefull or if the memory of the pc is full it could be necessary to unload some data. Pressing the button labeled **Unload test** (Figure 4.3) the software unload one or more selected dataset from the Node list. It's necessary that the user select one or more root in order to perform this task. After this operation the tree is updated without the unloaded data.

4.2.1.3 Add Data to New Plot

In order to view data in a plot to analyze them Lama View let the user to open a Lama Plot window (See Section 4.3 [Plot], page 13, for details). Pressing the button labeled **Add Data to New Plot** (Figure 4.3) Lama View opens new Lama Plot window with data selected in the Node List. If the user selects one node which one or more children all of them are plotted in the new window.
After this operation Lama View update the Windows List tree adding the new opened Lama Plot window.

4.2.1.4 Add data to an Existing Plot

Observing one plot it could be usefull compare data shown in the plot window with other data. Pressing the button labeled **Add data to an Existing Plot** (Figure 4.3) Lama View opens the list of opened windows. By pressing the desired window Lama View adds the selected nodes in the Node List to the window.

4.2.1.5 The Default Selection

The IDL procedures and functions (See Chapter 6 [IDL], page 25, for details) need a default test, a default feed horn and a default processing type. Pressing the button labeled **Make Default** (Figure 4.3) Lama View signs as default the selected node in the Node List. The default nodes are bold in the Node List.

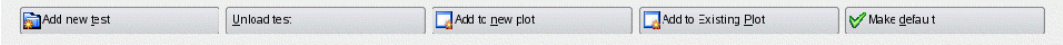

Figure 4.3: Buttons in Lama View main window

4.2.1.6 Right Mouse Button Menu

By clicking the right mouse button on the Node List a menu appears. The menu has five possible tasks to let the user to see information about files in the dataset folder, load or unload files, export data to file and open a cross-correlation plot.

- Open object information. One or more fits files are associated to each terminal node of the Node List. To see information about those files select one or more nodes in the Node List, click right mouse button and select Open Object Information. A new window is opened in which there is the view with the selected nodes and the information about the selected fits files, with information about theirs contents and if they are loaded or not.
- Load data. Select one or more nodes in the Node List, click the right mouse button and select Load Data. Lama View load the fits files associated to the selected nodes. After this operation the loaded nodes are in italics character.
- Unload data. Select one or more nodes in the Node List, click the right mouse button and select Unload Data. Lama View unload the fits files associated to the selected nodes. After this operation the unloaded nodes returns to normal character.
- Export data to file. Select one or more nodes in the Node List, click the right mouse button and select Export data to file. Lama View opens new window in which the user can select the name of file and the place where it must be saved. The file format is CSV ("Comma Separated Values"), a text file in which data are stored in the same row separated by commas. It's the simpler text format that can be imported in Excel or OpenCalc.
- Cross correlation plot. Select two nodes in the Node List, click the right mouse button and select Cross-correlate plot. Lama View load the fits files associated to the selected nodes, if necessary, and then open a new Cross correlation plot that shows the cross-correlation plot between the two selected nodes (See Section 4.4 [Cross], page 20, for details).

4.2.2 Window List

The Window List shows the list of all plot windows opened (either Lama Plot, either Lama Cross), with theirs titles (Figure 4.4). Expanding one node of the list Lama View shows the content of the expanded node.

- Bring to front. Selecting one window from the Window List and clicking the Bring to front button the selected window is brought to the front of the desktop.
- Combine windows. Selecting two or more windows from the Window List and clicking the Combine windows button the selected windows are closed and the data contained in all the windows are shown in a new plot window. The selected windows disappear from the Window List and a new item appear that represents the new window.
- Close window. Selecting one or more windows from the Window List and clicking the Close window button the selected windows are closed and disappear from the Window List.

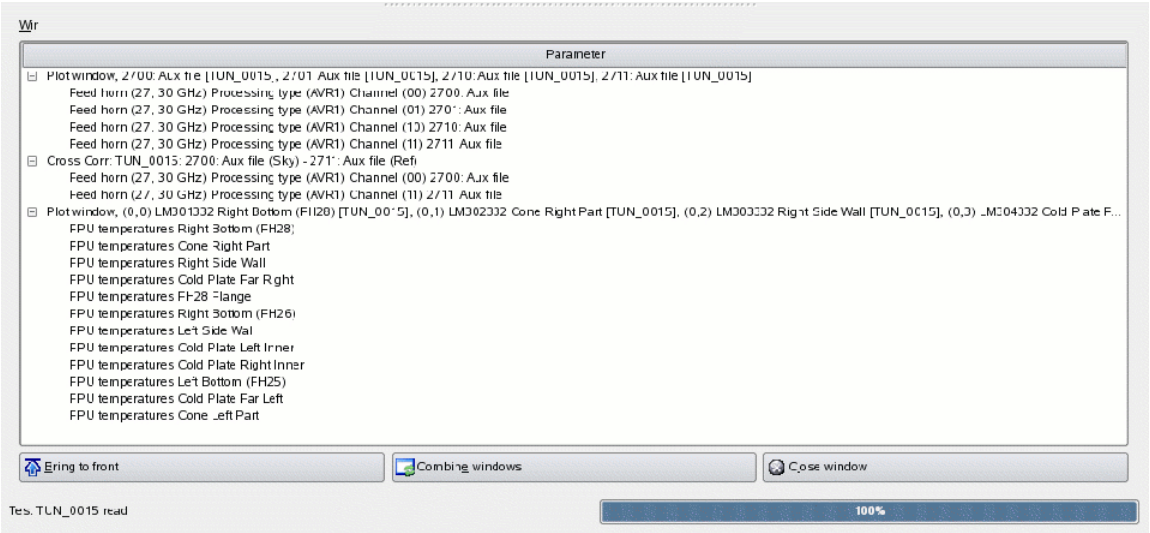
Parameter
Feed horn (27, 30 GHz) Processing type (AVR1) Channel (01) 270
Feed horn (27, 30 GHz) Processing type (AVR1) Channel (10) 2710: Aux file
Feed horn (27, 30 GHz) Processing type (AVR1) Channel (11) 2711 Aux file
Cross Corr: TUN_0015: 2700: Aux file (Sky) - 271
Feed horn (27, 30 GHz) Processing type (AVR1) Channel (00) 2700: Aux file
FPU temperatures Right Bottom (FH28)
FPU temperatures Cone Right Part
FPU temperatures Right Side Wall
FPU temperatures Cold Plate Far Right
FPU temperatures FH28 Flange
FPU temperatures Right Bottom (FH26)
FPU temperatures Left Side Wall
FPU temperatures Cold Plate Left Inner
FPU temperatures Cold Plate Right Inner
FPU temperatures Left Bottom (FH25)
FPU temperatures Cold Plate Far Left
FPU temperatures Cone Left Part
Bring to front
Combine windows
Close window
100%

Figure 4.4: Windows List in Lama View main window

4.3 The Lama Plot Window

Lama Plot window plots scientific and housekeeping values.

4.3.1 The plot

The plot is the image in the left part of Lama Plot (Figure 4.5). The image shows values of data selected in Lama View Node List view in different colors. The x-axis is ususally time, instead the y-axis is different for different types of data (e.g. all scientific data are in volts). In the bar at the top of the window there are different buttons for different operations (Figure 4.6). The next sections show how to use those tasks.

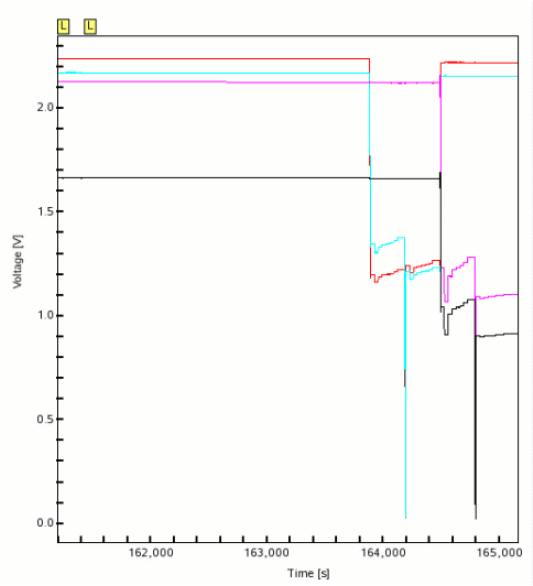

Figure 4.5: The plot

4.3.1.1 Zoom data

When the Zoom button (Figure 4.6) is selected it's possible to "zoom in" in the plot by clicking the left mouse button. Moving mouse with the left mouse pressed a rectangle is drawn in the window. After the release of the button Lama Plot updates the plot drawing data contained in the rectangle. If the user presses the right mouse button and moves the mouse Lama Plot draws a rectangle in the plot. After the release of the button Lama Plot updates the plot "zooming out" with, in the center, the content of the rectangle.

4.3.1.2 Navigating into the data

Using the Navigation buttons (Figure 4.6) it's possible to navigate backward and forward in the different zooms applied by the user to the plot.

Note that columns mean and sigma in the data information box will show information which is related with the portion of the plot shown in the plot window. Zooming in or out in the plot will make these values change.

4.3.1.3 Select data

When the Select button is selected (Figure 4.6) it's possible to select data into the plot by pressing the right mouse button and moving it. After the release Lama Plot stores the begin and the end of the x-range selected. This operation is possible only for scientific data. The next operation is to load selected data (see [ScientificSelection], page 16 for details).

4.3.1.4 Pan plot

When the Pan button is selected (Figure 4.6) it's possible to pan the plot. If the right mouse button is pressed when the user moves the mouse into the plot then the plot is moved. After the mouse release the plot is in the last position.

4.3.1.5 Reset plot

Clicking the button labeled Reset (Figure 4.6)will undo all the operations performed on the data and reset the zoom list. After clicking on this button the navigation buttons will be deactivated until another zoom operation is performed.

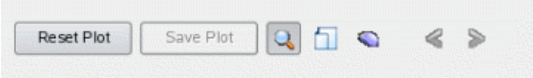

Figure 4.6: The buttons for plot interactions

4.3.1.6 The Plot Legend

In the right corner of the Lama Plot window there is the Legend (Figure 4.7). In the Legend there is one row for each data stream plotted with its name, its color, its style and a boolean which says if it's visible.
It's possible to select some data stream in the legend. Pressing the right mouse button a menu appears. The menu has the possibility to change the style or the color, to show or to hide a stream, to remove a stream, to open a crosscorrelation plot (see Section 4.4 [Cross], page 20 for details).

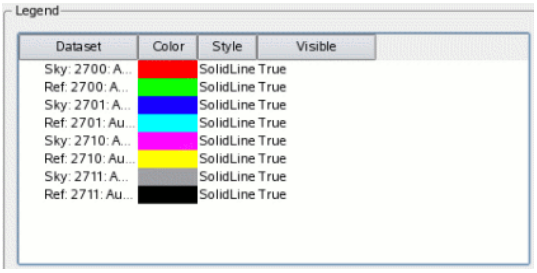

Figure 4.7: The Legend of the plot

4.3.2 Operations

The first tab in the right side of Lama Plot is the Operations tab (Figure 4.8). It let to remove mean from the plot and to normalize data in the plot. Those operation let a comparison between data plotted if they have a large range in the y-axis.

- Remove Mean. Pressing the Remove Mean button Lama Plot updates the plot removing the mean of each node plotted in the image.
- Normalize Data. Pressing the Normalize Data button Lama Plot updates the plot drawing each node plotted in the image in a range between -1 and +1.

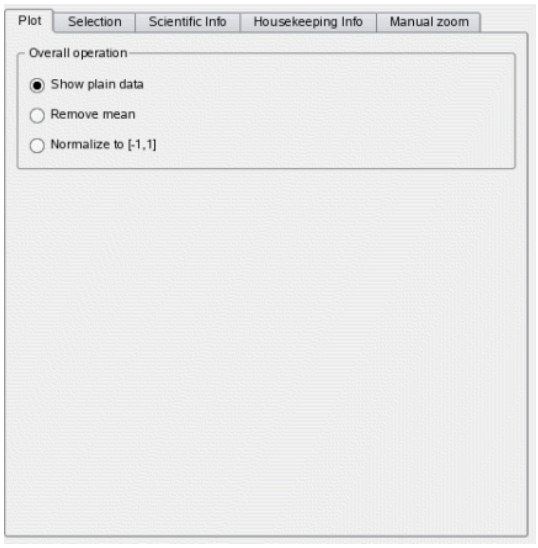

Figure 4.8: The plot operations tab

4.3.3 Scientific Selections

The second tab in right side of Lama Plot let the user to load scientific data (Figure 4.9). The data plotted in Lama Plot are the values contained in AUX files in which the scientific data are binned at the frequency of 1 Hz. If the user needs to see the complete values has to load them from the scientific files.

- Pressing the Load Data button Lama Plot loads values from the scientific fits files in the range selected in the plot with the choosen bin value and append them to the Node List. After this operation the scientific nodes in the Node List appeats in italic character.

- Pressing the Unload Data button Lama Plot unloads the scientific fits files removing them from the Node List. After this operation the scientific nodes in the Node List appeats in normal character.

- Pressing the Show Data button Lama Plot shows the loaded scientific data in a subplot. After this operation the Show Data button changes its name in Hide Data. If the user presses the button another times Lama Plot hide the subplot and brings back the name to Show Data.

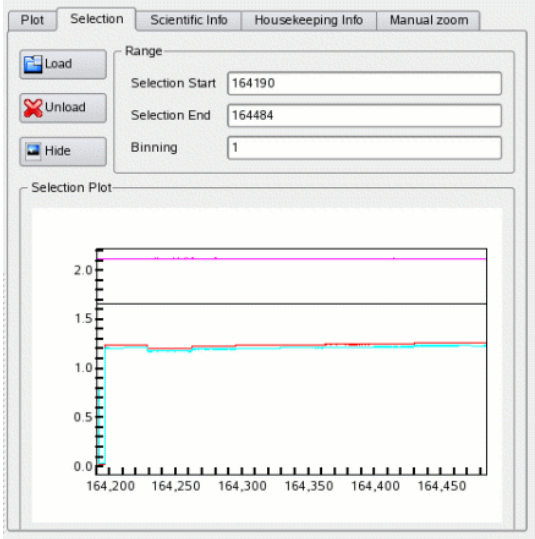

Figure 4.9: The plot selection tab

4.3.4 Radiometric Operations

The third tab in right side of Lama Plot shows the radiometric statistics of the plotted data (Figure 4.10). The list contains 6 columns. The first column is the name of the channel and the file (AUX or Scientific). The second and the thrid columns show the mean and the sigma values of the sky data stream, the fourth and the fifth the mean and the sigma values of the reference data stream. The last column is boolean the shows if the sky and reference data stream are swaped of not. Note that if the plot is zoomed the statistics are referred to this.

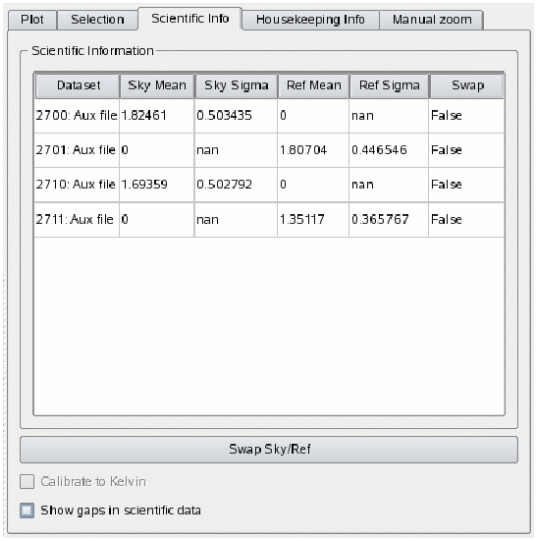

Figure 4.10: The plot operations tab

4.3.4.1 Swap sky and reference

If the DAE inverts sky and reference values those values are stored in fits files in wrong way. Lama shows data as they are stored in the fits files, so, in this case, the data shown are wrong. The Swap sky/ref button lets the user to swap sky and reference in the plot. Select the channel from the list of radiometric data and click Swap sky/ref button. After this operation the last column change from false to true and the background of the row become grey. Clicking again the button reset sky and reference in the plot and the row in the list of radiometric data.

4.3.4.2 Show gaps in scientific data

During operations DAE (or REBA) could loose some packets. Pressing the Show Gaps button let the user to know it this is happened.

4.3.5 Housekeeping information

The fourth tab in right side of Lama Plot shows the housekeeping statistics of the plotted data (Figure 4.11). The list in this tab contains three columns. The first column is the name of the housekeeping parameter. The second and the third columns show mean and sigma values of the parameter in the actual plot. Note that if the plot is zoomed the statistics are referred to this.

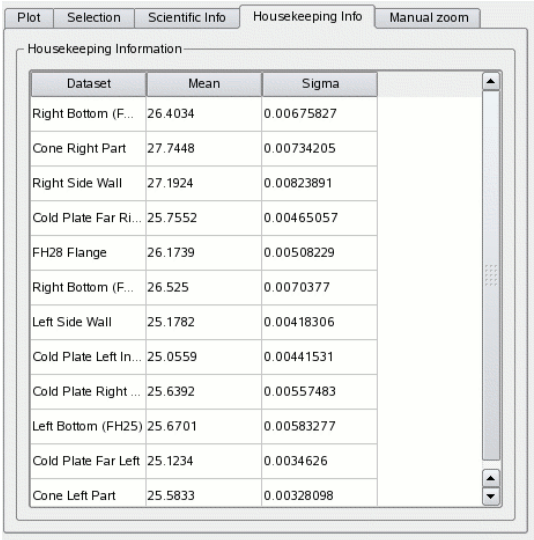

Figure 4.11: The plot housekeeping information tab

4.3.6 Manual operations

The last tab in the right side of Lama Plot shows the manual operation (Figure 4.12). There are two possible operations:

- There are four cells. Set one (or more) of those cells and press the button labeled **Zoom** shows the data in the selected range of zoom.

- Clicking the button labeled **SET** the two cells labeled **X Origin** and **X End** are updated with the range of the x axis. This range can be used for computing statistics available from IDL command line. See [statistics], page 31.

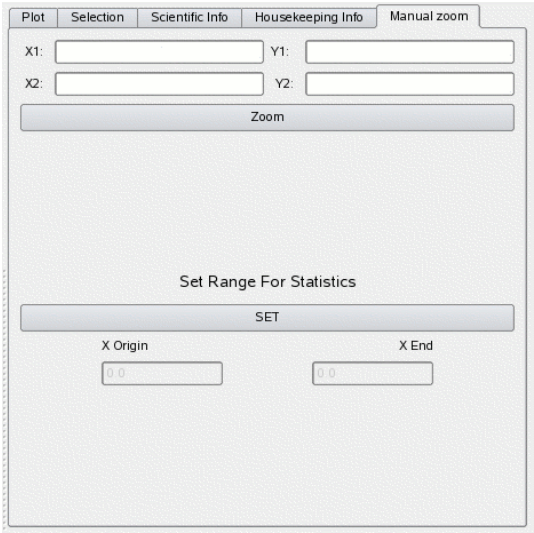

Figure 4.12: The plot manual operations tab

4.4 The Lama Cross-correlation Window

4.4.1 Choosing Data

There are two ways to display a cross correlation plot:

1. *Display a cross correlation plot form Lama View:* On the data tree select 2 timelines to be correlated, right click and choose *Cross Correlate Plot on the menu.*
2. *Display a cross correlation plot from the Plot Window:* On the legend select the two timelines to cross correlate, right click on the legend and choose *Cross Correlate Plot.*

Any of these actions starts the same cross correlation plot window. There however a few differences in the type of data that can be choosen. When a cross correlation plot is displayed from the Lama View window only cross correlation plots involving the sky signal can be created since the Lama Node List does not allow to choose between Sky and Reference for each channel from Lama View.

On the other hand, opening a cross correlation plot from the legend in the plot window will not only allow for sky or reference to be choosen but also to perform the cross correlation on a selected chunk of data: the cross correlation plot will only contain data displayed in the plot window. To display the cross correlation for a particular segment of data we can first zoom in on the relevant region in the Lama Plot window and then select the cross correlation plot.

It is therefore recommended that a cross correlation plot should start almost always from the plot window.

4.4.2 The plot

The plot is shown on the plot window which occupies most of the cross correlation window. the leftmost part of the cross correlation window.

4.4.2.1 Navigating into the data

The zoom button activates the zoom function. To zoom in on a particular region in the plot the procedure is exactely the same as for the Lama Plot window. The navigation buttons also work in the same way as those for the Lama Plot window.

Note that columns x,y and z in the data information box will show information which is related with the portion of the plot shown in the plot window. Zooming in or out on the plot will make these values change.

NOTE: The zoom button will probably be removed in the near future since there is no other navigation mode available for the cross correlation window.

4.4.2.2 Save data

The Save Data button will save data to an ASCII file. To save data click the Save Data button and select a directory where you would like the data file to be stored. The name of the file will be automatically generated from the X and Y data in the plot. There are two different formats for the ASCII file generated by pressing the Save Data button:

1. *Unbinned data* If the Bin button is not checked the file will contain two columns, the first containing X data and the second Y data. Note that this is not necessarily the actual raw data as some binning might have already ocurred to deal for instance with different sampling rates for the X and Y timestreams.
2. *Binned data* If the Bin button is checked the file will contain five columns: the first contains X corresponding to the center of the bins; the second the average of all X values contributing to the bin; the third the error associated with the bin average in column 2; the fourth the binned valued of Y and the fifth and last the error associated with the binned value for Y. The errors saved on the file are simply the standard deviation calulated from all the values in the bin.

   The files containing data with associated error bars have a _BIN appended at the end of the name.

4.4.3 Time Span

There are three numbers on the right upper corner immediately above the plot region:

1. *Min* The time associated with the first sample used in the cross correlation
2. *Max* The time associated with the last sample used in the cross correlation
3. *Range* The time interval associated with the full timelines used in the cross correlation

4.4.4 Operations

4.4.4.1 Reset

Clicking the button labeled Reset will undo all the operations on the data and reset the zoom list. After clicking on this button the navigation buttons will be deactivated until another zoom operation is performed.

4.4.4.2 Swap Axes

Clicking on the check box labelled Swap X/Y will swap the axes. Before displaying the plot the data is sorted according to the values originaly displayed as Y. Clicking one more time will display the original plot.

It is not possible to swap the axes when error bars are displayed.Conversely when the axis are swaped it is also not possible to show error bars.

4.4.4.3 Show points

Clicking on the check box labeled Show Data Points will display the data points. Note that when the cross correlation plot being displayed involves data streams with different sampling rates or when the sampling rate is the same for the two data streams but the sampling times are not synchronised some binning will automatically be performed and therefore the points displayed are not necessarily the real data points. Cliking again to uncheck the box will return to the line plot. The same action is obtained from the legend pop up menu. Selecting Change Style->Points from the menu when selecting a plot in the legend will also show the data points.

4.4.4.4 Show error bars

To show error bars in the plot click on the Show Error Bars check button after choosing which error bars to dsiplay using the radio buttons labeled X and Y. It is not possible to display error bars before binning the data. Cliking one more time will hide the error bars.

As mentioned before (See [Swap Axes], page 21.) it is not possible to display error bars when the axes are swaped.

4.4.4.5 Bin data

There are two ways to bin data

1. Choose the number of points you want on the binned plot by clicking on the Bin Count radio button, entering the number of points in the box and checking the check button labeled Bin. This will produce regularly distributed points.
2. Choose the number of points you want on each bin by checking the radio button labeled Bin Size and proced as in 1). This will produce irregularly distributed points in the plot.

To change the number of bins/data points per bin change the number in the edit box and click on bin *again*.

4.4.4.6 Show fit

The cross correlation module calculates the correlation coefficient as well as a simple linear least squares fit ($y=a+bx$) for the data displayed on the plot window. To display the fit check the Show Fit check box. To hide the fit click on the check box again.

4.4.5 Data Information Window

The data information window has 8 columns

1. Description of X data.
2. Description of Y data.
3. Correlation coefficient for the full data set or the region of the dataset originally selected from Lama Plot.
4. Correlation coefficient for the region currentely displayed in the correlation plot. This number changes after a zoom operation.

5. Linear fit coefficient a for the full dataset or the region of the dataset originally selected from Lama Plot.
6. Linear fit coefficient a for the region currentely displayed in the correlation plot.
7. Linear fit coefficient b for the full dataset or the region of the dataset originally selected from Lama Plot.
8. Linear fit coefficient b for the region currentely displayed in the correlation plot.

5 OCA

6 IDL Command Reference

This chapter provides a broad and fairly complete overview of the commands implemented by LIFE which are available through the IDL command line.

6.1 The Lama Instance

Lama uses a more complex design than RaNA. It uses a mix of IDL/C++ code to achieve greater speed and provide the user with a complete graphical user interface.

In order to use C++ modules, you have to create an "instance", that is, a data structure which keeps track of the various data structure that need to be shared both by IDL and by the C++ functions. See [lama_instance_create], page 25. Only one instance of Lama is needed during the program execution: typically, it is created when the "Start session" button in the Lama Top window is pressed, and destroyed when the "Close session" button is pressed.

`lama_instance_create` [*directory*] [Function]

Create a new instance of LAMA. This function needs to be called once, usually when the "Start session" button in the Lama Top window is pressed. The *directory* parameter is a string containing the full path to the '`life/lama`' directory, and is used to locate Lama-specific files like '`lama_parameters.xml`'. If *directory* is not specified, the current working directory will be used.

The return value is nonzero if the instance was created successfully, zero if some error occurred.

```
result = lama_instance_create ("/home/tomasi/work/life/lama")
```

`lama_instance_close` [Procedure]

Close the currently opened Lama instance. It is usually called when the user presses the "Close session" button in the Lama Top window.

`lama_instance_is_created` [Function]

This function returns nonzero if `lama_instance_create` was called and therefore the Lama system has been initialized.

Since almost every function in the Lama Input and Lama View modules requires a working Lama instance, this function is extremely useful in scripts to check if an instance has been created:

```
if ~lama_instance_is_created () then begin
    print, "Error: you need to create a Lama instance"
    exit, 1
endif
```

6.2 Opening and closing Lama View

`lama_view_open` [Function]

This function opens the Lama View window. If the window is already opened, it does nothing.

This function always returns a nonzero value.

`lama_view_close` [Function]
Close the Lama View window. If the windows is not open, does nothing.

`lama_view_is_opened` [Function]
Return nonzero if the Lama View window has been opened (i.e. the [lama_view_open], page 25 function has been called).

6.3 Loading/unloading tests

`lama_input_load_test` *dir* [Function]
This function opens the test found in the directory named *dir* (a string), and marks it as the new default test. If it is successful, returns a nonzero value, otherwise returns zero.

The *dir* variable can contain either an absolute path, or a path relative to the current working directory. However, in scripts the absolute path should be preferred.

```
test_to_load = "/home/tomasi/work/datasets/TUN_0032"
result = lama_input_load_test (test_to_load)
if result eq 0 then begin
    print, "Unable to open the file located in ", test_to_load
end if
```

`lama_input_unload_test` *name* [Procedure]
This function remove test named *name* from memory. Note that *name* must be the name of the test, not the full directory. See [lama_input_get_list_of_tests], page 27.

`lama_input_is_test_available` [Function]
This function returns nonzero if at least one test has been loaded. See [lama_input_load_test], page 26.

`lama_input_select_test` *name* [Function]
This function makes *name* the default test. Every time a function which provides the `TEST` keyword (e.g. [lama_get_sky_y], page 29) is called without specifying it, the test passed to `lama_input_select_test` will be used.

The function will return true (nonzero) if the test was selected, false if the specified test does not exists (i.e. it has not been loaded).

name must be one of the strings returned by [lama_input_get_list_of_tests], page 27.

`lama_input_get_selected_test` [Function]
Return a string containing the name of the currently selected test. Here is an example of use:

```
;; Save the currently selected test
saved_test = lama_input_get_selected_test ()

;; Load a new test
lama_input_load_test ("/home/tomasi/work/datasets/XXX_0012")
```

```
;; Do whatever you want with XXX_0012
...

;; Now select again the old test
lama_input_select_test, saved_test
```

`lama_input_get_list_of_tests` [Function]

Return an array of strings containing the currently loaded tests. Each string contains only the test name, not the full path. You can use any of them with [lama_input_select_test], page 26 and [lama_input_unload_test], page 26: this is guaranteed to work (if it does not, it is a bug!).

```
;; This code will select the first tests as the default

tests_loaded = lama_input_get_list_of_tests ()
if ~lama_input_select_test (tests_loaded[0]) then begin
    print, "Error: unable to load test ", tests_loaded[0]
endif
```

6.4 Feed horns

`lama_input_select_feed_horn` *feed TEST=test* [Function]

Set *feed* to be the default feed horn for test named *test*. If the `TEST` keyword it is not specified, the default test will be used. See [lama_input_select_test], page 26.

The *feed* variable can be specified in three ways:

1. A string with the feed horn number alone, e.g. `'28'`;
2. A string with the pound character `#` followed by the feed horn number, e.g. `'#28'`;
3. The number of the feed horn, e.g. `28`. This is useful when you are selecting horns in a `for` loop.

The function will return true (i.e. nonzero) if the feed horn was selected, false if the feed horn does not exists or its output has not been recorded by SCOS during the specified test.

```
;; This code will cycle through all the 11 feed horns
for i = 18, 28 do begin
    if ~ lama_input_select_feed_horn (i) then begin
        print, "Error with feed horn ", i, ", skipping"
    endif else begin
        ;; Do some operation on feed horn number I
    endelse
endfor
```

`lama_input_get_selected_feed_horn` *feed TEST=test* [Function]

This function will return the selected feed horn for test named *test*, or for the default test if `TEST` was not specified. See [lama_input_select_test], page 26.

This function is guaranteed to return a value which is usable by [lama_input_select_feed_horn], page 27.

6.5 Accessing data

In this section we provide the documentation for functions accessing the scientific/housekeeping data. With "scientific data" we mean the samples coming out from the radiometers, while "housekeeping data" include everything other stream of data recorded by SCOS2000 (i.e. drain currents, temperatures and so on).

6.5.1 Scientific data

The functions listed in this sections let to access the scientific data coming from the radiometers. You must know there are two kind of scientific data available with Lama:

Full scientific data
: This is the data coming from the radiometers. Its sampling frequency depends on the radiometer (being greater for the 70 GHz ones) and the REBA processing type, but it is never smaller than 1 Hz.

Binned data
: This is the default type of data loaded by Lama View. Its sampling frequency is always 1 Hz, regardless of the radiometer and the processing type, and it is calculated from full scientific data by evaluating the mean value and the standard deviation in a time span of 1 second per each binned sample.

Binned data is something that is not produced by the TQL/TMH software: the binned datasets are created by Lama "on the fly", in order to reduce the time needed to load and display data. The FITS files containing binned data are called "AUX" files.

6.5.1.1 General functions

`lama_get_num_of_sci_tabs` *TEST=test* [Function]
: Return the overall number of scientific channels present in *test*. If the `TEST` keyword was not specified, use the default test.

In some cases the acquisition system is not able to identify the sky and reference signals correctly. Lama provides the ability to swap the two signals by means of [lama_swap_sky_ref], page 28. You can determine if a channel has been already swapped by using the function [lama_channel_is_swapped], page 28.

`lama_swap_sky_ref` *channel TEST=test FEED_HORN=feed_horn* [Procedure]
: Swap sky and reference values. If no keywords is specified do nothing. If `CHANNEL` is specified swap sky and reference of that channel of the default test of the default feed_horn. If `FEED_HORN` is specified swap sky and reference of all channels of the *feed_horn* of the default test. If `TEST` is specified swap all horns of the *test*.

`lama_channel_is_swapped` *channel_num TEST=test_name* [Function]
FEED_HORN=feed_horn PROC_TYPE=proc_type
: Return 1 if the scientific sky and reference signals coming from channel *ch* are swapped. If `TEST`, `FEED_HORN` and `PROC_TYPE` are not specified, the defaults are used.

6.5.1.2 Accessing binned data

The following functions give access to the data saved in the AUX files, i.e. the scientific data downsampled at 1 Hz.

`lama_get_sky_x` *ch TEST=test FEED_HORN=horn* [Function]
PROC_TYPE=proc_type UNIT=unit

Return the time series for the scientific binned sky signal coming from channel *ch*. If `TEST`, `FEED_HORN` and `PROC_TYPE` are not specified, the defaults are used. If `UNIT` is specified return the series in the specified measure unit, otherwise return time in seconds. See [measureUnits], page 45.

`lama_get_ref_x` *ch TEST=test FEED_HORN=horn* [Function]
PROC_TYPE=proc_type UNIT=unit

Return the time series for the scientific binned reference signal coming from channel *ch*. See [lama_get_sky_x], page 29.

`lama_get_raw_x` *ch TEST=test FEED_HORN=horn* [Function]
PROC_TYPE=proc_type, UNIT=unit

Return the time series for the scientific binned reference signal coming from channel *ch*. The sky and reference samples are alternating, in order to mimic the way they come from the BEU. See [lama_get_sky_x], page 29.

`lama_get_sky_y` *ch TEST=test FEED_HORN=horn* [Function]
PROC_TYPE=proc_type UNIT=unit PGA_CORRECT=pga RAW=raw

Return the data series for the scientific binned sky signal coming from channel *ch*.

If *no_ofs* is nonzero, the values will have their mean removed from the data (i.e. they will have zero mean).

The `PGA_CORRECT` and `RAW` keywords are implemented for compatibility with RaNA. See [measureUnits], page 45.

`lama_get_sky_sigma` *ch TEST=test FEED_HORN=horn* [Function]
PROC_TYPE=proc_type UNIT=unit PGA_CORRECT=pga RAW=raw

Return the standard deviation (i.e. the root mean square) of each binned sky sample. See [lama_get_sky_y], page 29.

`lama_get_ref_y` *ch TEST=test FEED_HORN=horn* [Function]
PROC_TYPE=proc_type UNIT=unit PGA_CORRECT=pga RAW=raw

Return the data series for the scientific binned reference signal coming from channel *ch*. See [lama_get_sky_y], page 29.

`lama_get_ref_sigma` *ch TEST=test FEED_HORN=horn* [Function]
PROC_TYPE=proc_type UNIT=unit PGA_CORRECT=pga RAW=raw

Return the standard deviation (i.e. the root mean square) of each binned ref sample. See [lama_get_ref_y], page 29.

`lama_get_raw_y` *ch TEST=test FEED_HORN=horn* [Function]
PROC_TYPE=proc_type UNIT=unit PGA_CORRECT=pga RAW=raw
Return the data series for the scientific binned reference signal coming from channel *ch*. The sky and reference samples are alternating, in order to mimic the way they come from the BEU. See [lama_get_sky_y], page 29.

`lama_get_raw_sigma` *ch TEST=test FEED_HORN=horn* [Function]
PROC_TYPE=proc_type UNIT=unit PGA_CORRECT=pga RAW=raw
Return the standard deviation (i.e. the root mean square) of each binned sky/ref sample. The sky and reference samples are alternating, in order to mimic the way they come from the BEU. See [lama_get_raw_y], page 29.

6.5.1.3 Accessing full scientific data

The following functions provide access to the full scientific data. To be able to access this data, the user must specify a time range where to load the data and a binning. The frequency sampling of scientific data depends on the acquisition mode and the specified binning.

To load the full scientific data, use the [lama_select_range], page 30 procedure.

`lama_select_range` *ch min max bin TEST=name HORN=horn* [Procedure]
PROC_TYPE=proc_type
Load scientific data inside the specified time range [*min*, *max*] (in seconds) and using *bin* samples for the binning (i.e. *bin* equal to 1 means that there is no binning). If `TEST`, `FEED_HORN` and `PROC_TYPE` are not specified, the defaults are used.

In order to determine if some scientific selection has been loaded, you can use either [lama_selection_is_sky_available], page 30 and [lama_selection_is_ref_available], page 30. The two functions are equivalent under Lama.

`lama_selection_is_sky_available` *ch TEST=name HORN=horn* [Function]
PROC_TYPE=proc_type
Return nonzero if the scientific selection loaded by the last call to [lama_select_range], page 30 contains a non-void stream of sky data.

`lama_selection_is_ref_available` *ch TEST=name HORN=horn* [Function]
PROC_TYPE=proc_type
Return nonzero if the scientific selection loaded by the last call to [lama_select_range], page 30 contains a non-void stream of sky data.

`lama_selection_get_sky_x` *ch TEST=test FEED_HORN=horn* [Function]
PROC_TYPE=proc_type UNIT=unit
Return the time series for the scientific binned sky signal coming from channel *ch*. If `TEST`, `FEED_HORN` and `PROC_TYPE` are not specified, the defaults are used. If `UNIT` is specified return the series in the specified measure unit, otherwise return time in seconds. See [measureUnits], page 45.

`lama_selection_get_ref_x` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type UNIT=unit* [Function]

Return the time series for the scientific binned reference signal coming from channel *ch*. See [lama_selection_get_sky_x], page 30.

`lama_selection_get_raw_x` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type UNIT=unit* [Function]

Return the time series for the scientific binned sky/reference signal coming from channel *ch*. The sky and reference samples are alternating, in order to mimic the way they come from the BEU. See [lama_selection_get_sky_x], page 30.

`lama_selection_get_sky_y` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type UNIT=unit PGA_CORRECT=pga RAW=raw* [Function]

Return the data series for the scientific binned sky signal coming from channel *ch*.

For a discussion about `UNIT`, `PGA_CORRECT` and `RAW` see [lama_get_sky_y], page 29.

`lama_selection_get_sky_sigma` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type UNIT=unit PGA_CORRECT=pga RAW=raw* [Function]

Return the standard deviation (i.e. the root mean square) of each scientific sky sample. See [lama_selection_get_sky_y], page 31.

`lama_selection_get_ref_y` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type UNIT=unit PGA_CORRECT=pga RAW=raw* [Function]

Return the data series for the scientific binned reference signal coming from channel *ch*. See [lama_selection_get_sky_y], page 31.

`lama_selection_get_ref_sigma` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type UNIT=unit PGA_CORRECT=pga RAW=raw* [Function]

Return the standard deviation (i.e. the root mean square) of each scientific reference sample. See [lama_selection_get_ref_y], page 31.

`lama_selection_get_raw_y` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type UNIT=unit PGA_CORRECT=pga RAW=raw* [Function]

Return the data series for the scientific binned sky/reference signal coming from channel *ch*. The sky and reference samples are alternating, in order to mimic the way they come from the BEU. See [lama_selection_get_sky_y], page 31.

`lama_selection_get_raw_sigma` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type UNIT=unit PGA_CORRECT=pga RAW=raw* [Function]

Return the standard deviation (i.e. the root mean square) of each scientific sky/reference sample. The sky and reference samples are alternating, in order to mimic the way they come from the BEU. See [lama_selection_get_raw_y], page 31.

6.5.1.4 Statistics

Lama provides the user with some functions to perform quick analysis on the selected data. They are useful e.g. for writing batch scripts.

```
range = lama_selection_get_range ()
average = lama_selection_get_sci_ave (0)
sigma = lama_selection_get_sci_sigma (0)

print, "Between ", range[0], " s and ", range[1], $
       " s the sky signal is"
print, average[0] " +/- ", sigma[0], " and the ref. signal is"
print, average[1] " +/- ", sigma[1]
print, "The difference is then:"
print, abs (average[0] - average[1]), " +/- ", $
       sqrt (sigma[0]^2 + sigma[1]^2)
```

`lama_get_sci_nsamples` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type* [Function]

Return an array (of the same size as the ones returned by [lama_get_sky_x], page 29, [lama_get_sky_y], page 29, [lama_get_ref_x], page 29 and [lama_get_ref_y], page 29) containing the number of scientific samples used to create the aux binned data. If `TEST`, `FEED_HORN` and `PROC_TYPE` are not specified, the defaults are used.

`lama_get_sci_ave` *ch range_min=[min_time] range_max=[max_time] TEST=test FEED_HORN=horn PROC_TYPE=proc_type X_UNIT=x_unit UNIT=y_unit* [Function]

Return the mean of the selected scientific/reference data for channel *ch*. If *min_time* and *max_time* are specified, then the mean is calculated in the specified time window. Otherwise, it is calculated in the current statistical range (see [lama_get_statistic_range], page 33). The result is a two-valued array where the first value refers to the sky, the second to the reference load. If `X_UNIT` keyword is not specified then the function uses seconds for the time series. If `UNIT` is not specified then the function returns values in Volt. See [measureUnits], page 45.

`lama_get_sci_sig` *ch range_min=[min_time] range_max=[max_time] TEST=test FEED_HORN=horn PROC_TYPE=proc_type X_UNIT=x_unit UNIT=y_unit* [Function]

Return the standard deviation of the selected scientific/reference data for channel *ch*. If *min_time* and *max_time* are specified, then the mean is calculated in the specified time window. Otherwise, it is calculated in the current statistical range (see [lama_get_statistic_range], page 33). The result is a two-valued array where the first value refers to the sky, the second to the reference load. If `X_UNIT` keyword is not specified then the function uses seconds for time series. If `UNIT` is not specified then the function returns values in Volt. See [measureUnits], page 45.

`lama_selection_get_binning` *TEST=test FEED_HORN=horn PROC_TYPE=proc_type* [Function]

Return the binning used in the last call to [lama_select_range], page 30 to load a selection of scientific data for the specified feed horn and processing type. If `TEST`, `FEED_HORN` and `PROC_TYPE` are not specified, the defaults are used.

`lama_selection_get_nsamples` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type* [Function]

Return the number of samples loaded during the last call to [lama_select_range], page 30 to load the scientific data for channel *ch*. If `TEST`, `FEED_HORN` and `PROC_TYPE` are not specified, the defaults are used.

`lama_selection_get_range` *TEST=test FEED_HORN=horn PROC_TYPE=proc_type* [Function]

Return the time range used in the last call to [lama_select_range], page 30. The result is a two-valued array containing the two time limits (in seconds) where scientific data has been selected.

`lama_selection_get_sci_avg` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type X_UNIT=x_measure_unit UNIT=y_measure_unit* [Function]

Return the mean of the selected scientific/reference data for channel *ch*. The result is a two-valued array where the first value refers to the sky, the second to the reference load. If `X_UNIT` keyword is not specified then the function uses seconds for time series. If `UNIT` is not specified then the function returns values in Volt. See [measureUnits], page 45.

`lama_selection_get_sci_sigma` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type X_UNIT=x_measure_unit UNIT=measure_unit* [Function]

Return the standard deviation of the selected scientific/reference data for channel *ch*. The result is a two-valued array where the first value refers to the sky, the second to the reference load. If `X_UNIT` keyword is not specified then the function uses seconds for time series. If `UNIT` is not specified then the function returns values in Volt. See [measureUnits], page 45.

`lama_is_statistic_range_selected` *TEST=test* [Function]

Return nonzero if the user has selected a range for computing statistics (through one of the Lama Plot windows). See [Manual Operations], page 19. If *test* is not specified, the default one will be used.

`lama_get_statistic_range` *TEST=test* [Function]

Return the range used by [lama_get_sci_ave], page 32 and [lama_get_sci_sig], page 32 to do statistics. If *test* is not specified, the default one will be used.

6.5.1.5 Photometric calibration

Lama has the ability to perform a photometric calibration to convert the voltages coming out from a radiometer into an antenna temperature. In order to do this, the user must supply a calibration curve per each channel and processing type. This curve will be saved into FITS files which will be reused by Life.

A calibration curve must be provided as a TOD, i.e. a stream of time-ordered data, using the [lama_set_calibration], page 34 function.

`lama_set_calibration` *ch time cal TEST=test FEED_HORN=horn PROC_TYPE=proc_type* [Function]

Set the photometric calibration curve for channel *ch*. If *test*, *horn* and *proc_type* are not specified, the defaults will be used. The *time* and *cal* arguments are two arrays with the same size containing the time (in seconds) and the calibration constant (in K/V) to be used respectively. The return value is nonzero if the calibration was set successfully and written on the disk, zero if some error occurred.

An example of use is:

```
time = [  32671., 32900. ]
cal  = [   1.031,  1.046 ]

result = lama_set_calibration (0, time, cal)
if result eq 0 then begin
   print, "Error: unable to set calibration for channel 0"
endif
```

This example will instruct Lama to use the photometric constant 1.031 from time 32671 to 32900, and the constant 1.046 from 32900 till the end of the scientific datastream. The function will also save this calibration curve into a FITS file, so that it will be no longer needed to specify the curve if the test is to be loaded again.

Note that if a channel has been sampled using two different acquisition modes (e.g. AVR1 and COM5) you have to provide *two* calibration curves.

To determine if a channel already has a calibration curve associated with its scientific data, use the [lama_is_channel_calibrated], page 34 function.

`lama_is_channel_calibrated` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type* [Function]

Return nonzero only if channel *ch* has a photometric calibration curve. This happens when one of the following two conditions are met:

1. The function [lama_set_calibration], page 34 has been called for this channel during the current working session;
2. There is a FITS file in the '`toi`' directory of the test named *test* (the default test if not specified) which contains a valid calibration curve for the specified channel.

6.5.1.6 Miscellanea

The following two functions are designed to get the path of a scientific node for further use with browsing functions, like [lama_input_get_node_description], page 41 (see section [IDLBrowse], page 38).

`lama_get_aux_path` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type* [Function]

Return the path of the AUX data node associated with channel *ch*, test *test*, feed horn *horn* and processing type *proc_type*.

`lama_get_selection_path` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type* [Function]

Return the path (see section [IDLBrowse], page 38) of the node associated with the scientific selection of channel *ch*, test *test*, feed horn *horn* and processing type *proc_type*. This path can be used with a number of browsing functions, like [lama_input_get_node_description], page 41.

Note that this function works even if no selection has been loaded yet for the specific channel. It simply returns a path to the position where the selection *would* be loaded.

The following functions are specializations of [lama_input_unload_node], page 43 and are useful when you have completed some elaboration on a feed horn or radiometric channel and want to free some memory. See also [lama_input_unload_hk], page 38. Refer to section [IDLUnloadData], page 43 for more information about unloading data from memory.

`lama_input_unload_feed_horn` *feed_horn_num TEST=test* [Procedure]

This function unloads any X and Y data associated with the feed horn number *feed_horn_num* (this includes any selection made through [lama_select_range], page 30). You can still call [lama_get_sky_x], page 29, [lama_get_sky_y], page 29, [lama_get_ref_x], page 29 and [lama_get_ref_y], page 29 after a call to this function: Lama will silently load the data again.

The following example shows how to save some memory when processing data from feed horns:

```
;; Retrieve sky data from feed horn #27 (30 GHz)
feed_horn = 27
x = lama_get_sky_x (feed_horn)
y = lama_get_sky_y (feed_horn)

;; We have the sky data available into the two IDL vectors X
;; and Y, so we can unload it from Lama View
lama_input_unload_feed_horn, feed_horn

;; Now use X and Y freely!
...
```

`lama_input_unload_aux` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type* [Procedure]

Free any memory associated with the AUX datastream for channel *ch*, test *test*, feed horn *horn* and processing type *proc_type*.

`lama_input_unload_selection` *ch TEST=test FEED_HORN=horn PROC_TYPE=proc_type* [Procedure]

Free any memory associated with the scientific selection for channel *ch*, test *test*, feed horn *horn* and processing type *proc_type*. Calling this function when there is no scientific solution does not harm: the function will silently returns with no error.

Note that, unlike [lama_select_range], page 30, this function will operate on *one* channel, rather than on all the scientific channels of a feed horn.

6.5.2 Housekeeping data

`lama_get_hk_name` *tab element* [*TEST=test*] [Function]

Return the name of an housekeeping parameter. It can be in the form `LMxxxxxx` with each `x` being a digit (for housekeeping parameters recorded by SCOS2000) or `#fm-XXX:nn` with `XXX` being a lowercase name and `nn` a number (for cryofacility temperatures).

`lama_get_hk_description` *name|tab* [*element*] [*TEST=test*] [Function]

Return the description of an housekeeping parameter. It is an human-readable description of the parameter (e.g. "Drain current for #2700"). For an explanation of the parameters of this function, see [lama_get_hk_x], page 36.

There are two functions to get the data associated with an housekeeping file. They are [lama_get_hk_x], page 36 and [lama_get_hk_y], page 36, and return the X data series (time) and the Y data series (whose meaning depends on the parameter being selected).

`lama_get_hk_x` *name|tab* [*element*] *TEST=test UNIT=measure_unit* [Function]

Return the X data stream (time) associated with the specified housekeeping parameter. If *test* is not specified, the default one will be used. If `UNIT` is not specified return time in seconds. See [measureUnits], page 45.

The function can be called in two ways:

1. When used with *one* parameter (*name*), you pass a string containing the `LM` code of the parameter;
2. When used with *two* parameters (*tab*, *element*), you pass the tab and element numbers of the housekeeping.

Tabs and elements are the remnants of the way old Lama 1.0 handled housekeepings, and should be avoided in new code. You should prefer to use the LM code, as it is guaranteed not to change between different tests.

```
;; This is the first form. The name refers to the drain current
;; of FH#27 M1
x1 = lama_get_hk_x ('LM051322')

;; And this is the second one. Note that the housekeeping specified
;; by the (1, 15) pair is not always the same, but depends on the
;; test currently loaded
x2 = lama_get_hk_x (1, 15)
```

`lama_get_hk_y` *name|tab* [*element*] *TEST=test UNIT=measure_unit* [Function]
Return the X data stream (time) associated with the specified housekeeping parameter. If *test* is not specified, the default one will be used. `UNIT` specifies the desired measure unit. See [measureUnits], page 45.

For an explanation of the parameters accepted by this function, see [lama_get_hk_x], page 36.

6.5.2.1 Housekeeping Statistics

`lama_get_env_ave` *ch hk_tab, hk_element, TEST=test,* [Function]
X_UNIT=x_measure_unit, Y_UNIT=y_measure_unit
Return the mean of the housekeeping parameter *hk_tab*, *hk_element*. The mean is calculated in the current statistical range (see [lama_get_statistic_range], page 33). If `X_UNIT` keyword is not specified then the function uses seconds for time serie. If `Y_UNIT` is not specified then the function uses RAW data for data serie. See [measureUnits], page 45.

`lama_get_env_sig` *ch hk_tab, hk_element, TEST=test,* [Function]
X_UNIT=x_measure_unit, Y_UNIT=y_measure_unit
Return the sigma of the housekeeping parameter *hk_tab*, *hk_element*. The mean is calculated in the current statistical range (see [lama_get_statistic_range], page 33). If `X_UNIT` keyword is not specified then the function uses seconds for time serie. If `Y_UNIT` is not specified then the function returns raw (uncalibrated) data. See [measureUnits], page 45.

6.5.2.2 Miscellanea

The following function is designed to get the path of an housekeeping node for further use with browsing functions, like [lama_input_get_node_description], page 41 (see section [IDLBrowse], page 38).

`lama_get_hk_path` *name|tab* [*element*] *TEST=test* [Function]
Return the path of the data node associated with the housekeeping parameter with name *name*, or identified by the two index numbers *tab* and *element*. A typical usage is:

```
path1 = lama_get_hk_path ('LM101326') ; New LM syntax - preferred
path2 = lama_get_hk_path (1, 5)       ; Old syntax - compatible with RaNA

print, 'Node 1 is ', lama_input_get_node_description (path1)
print, 'Node 2 is ', lama_input_get_node_description (path2)
```

The following procedure is used to get a list of all the available housekeeping parameters in the specified test.

`lama_dump_hk_tables` *TEST=test* [Procedure]
This procedure prints on the IDL terminal a list of all the housekeeping parameters found in *test*, or in the default test if the `TEST` keyword is not specified. For each parameter the following information is provided:

1. The pair to be used with [lama_get_hk_x], page 36 and [lama_get_hk_y], page 36;
2. The LM code of the parameter, to be used with [lama_get_hk_x], page 36 and [lama_get_hk_y], page 36 and as reference;
3. The full description of the parameter.

The following function is a specialization of [lama_input_unload_node], page 43. See also [lama_input_unload_feed_horn], page 35 and similar functions. Refer to section [IDLUnloadData], page 43 for more information about unloading data from memory.

`lama_input_unload_hk` *name TEST=test* [Procedure]

This function unloads the X and Y data from the housekeeping parameter identified by *name*. You can still call [lama_get_hk_x], page 36 and [lama_get_hk_y], page 36 after a call to this function: Lama will silently load the data again.

The following example shows how to save some memory when processing data from housekeeping parameters:

```
;; Retrieve data from an housekeeping parameter
hk_name = 'LM302122'
x = lama_get_hk_x (hk_name)
y = lama_get_hk_x (hk_name)

;; We have the housekeeping data available into the two IDL
;; vectors X and Y, so we can unload it from Lama View
lama_input_unload_hk, hk_name

;; Now use X and Y freely!
...
```

6.5.3 Browsing the node tree

Starting from Lama 2.0, a new approach to access test data has been developed. The housekeeping and scientific parameters recorded during a test are stored in a tree-like structure. The basic element of this tree is a "node", i.e. a piece of information. Nodes are connected hierarchically by means of parent/child relationships: each node has one *parent* node and several *child nodes*.

This is the view of some details of the full node tree used by Lama:

```
Scientific data
  30 GHz feed horns
    FH#27
      Processing type: AVR1
        Channel: 00
          Scientific node
          AUX node
          Calibration node
        Channel: 01
          Scientific node
          AUX node
          Calibration node
```

```
Temperatures

  FPU temperatures
    Housekeeping: Right Bottom (FH28)
    Housekeeping: Cone Right Part
    Housekeeping: Right Side Wall
    Housekeeping: Left Bottom (FH25)
    Housekeeping: Cold Plate Far Left
    Housekeeping: Cone Left Part

  BEU temperatures
    Housekeeping: L-BEM 1
    Housekeeping: L-BEM 2
```

This example should make the parent/child relationship clear. For instance, the "FPU temperatures" node has six children (all are housekeeping parameters), while its parent is the "Temperatures" node.

Nodes are divided into different types according to the information they provide:

Scientific nodes
: They contain the sky and reference signal coming from a specified channel and recorded using a specified processing type.

Housekeeping nodes
: They are used to store information about an housekeeping parameter monitored by SCOS2000.

AUX nodes
: These nodes contain the subsampled scientific data coming from a specified channel and processing type.

Calibration nodes
: These nodes contain the information needed to apply a photometric calibration on the data coming from a specified channel and processing type.

Parameter groups
: These nodes do not contain information, but are used to hierarchically group a set of similar nodes (e.g. the "FPU temperatures" node contains a set of housekeeping parameters used to monitor temperatures in the Front End).

Processing types
: These nodes are used to group together a set of channels belonging to the same feed horn.

Feed horns
: These nodes are used to group together a list of processing types.

6.5.3.1 Node paths

You can consider the node structure to be much like the layout of your hard drive: some kind of nodes behave like files (e.g. scientific, housekeeping), while others are like directories

(e.g. parameter groups). We used this analogy to develop an IDL interface to access the tree structure: to find or uniquely specify a node you have to provide a "path" to the node.

Node paths are made by concatenating the names of each parent nodes, following the hierarchical sequence, separated by a '&'. For instance, the full path to `Cone Right Part` is `Temperatures&FPU temperatures&Cone Right Part`. To get the node names you can use the function `lama_input_get_node_name`.

`lama_input_get_node_name` [*path*] *TEST=name* [Function]
Return the name of the node relative to *path* (a string) and belonging to the test named *name*. If *path* is not specified, the current node in the test will be used (see [current_node], page 42). If *name* is not specified, the current test will be used.

To traverse the node tree you need to know what are the children of a specified node. This is provided by the function `lama_input_get_children`.

`lama_input_get_children` [*path*] *TEST=name* [Function]
Return a string array containing the children of the node specified by *path* and belonging to the test named *name*. If *path* is not specified, the current node of the test will be used (see [current_node], page 42). If *name* is not specified, the default test will be used. If the node has no children, the function returns -1.

Here is an example showing how to print the full node tree by implementing a recursive IDL procedure.

```
pro print_node_tree, start_path, indent_level

    ;; Create a string made by a number of INDENT_LEVEL
    ;; whitespace characters
    if indent_level gt 0 then begin
        byte_arr = bytarr (indent_level)
        byte_arr[*] = byte (' ')
        indent = string (byte_arr)
    endif else indent = ""

    ;; Print the name of the node
    print, indent, lama_input_get_node_name (start_path)

    ;; Recursively iterate through children
    child_list = lama_input_get_children (start_path)
    if size (child_list, /n_dim) gt 0 then begin
        for i = 0, n_elements (child_list) - 1 do begin
            ;; Make up the path of this child node
            if start_path ne '&' then $
              child_path = start_path + '&' + child_list[i] $
            else $
              child_path = '&' + child_list[i]

            ;; Print the node tree of this child
            print_node_tree, child_path, indent_level + 2
```

```
            endfor
        endif
    end
```

There are additional functions which retrieve other information about a node. They are listed here.

`lama_input_get_node_description` [*path*] *TEST=name* [Function]
Return a description for the node with path *path* and belonging to the test named *name*. If *path* is not specified, the current node in the test will be used (see [current_node], page 42). If *name* is not specified, the current test will be used.

`lama_input_get_node_x_units` [*path*] *TEST=name* [Function]
Return an array of IDs containing the valid measure units to be used for the X data of node with path *path* and belonging to the test named *name*. If *path* is not specified, the current node in the test will be used (see [current_node], page 42). If *name* is not specified, the current test will be used.

If the node does not contain data (e.g. if it is a group node), -1 will be returned.

`lama_input_get_node_y_units` [*path*] *TEST=name* [Function]
Return an array of IDs containing the valid measure units to be used for the Y data of node with path *path* and belonging to the test named *name*. If *path* is not specified, the current node in the test will be used (see [current_node], page 42). If *name* is not specified, the current test will be used.

If the node does not contain data (e.g. if it is a group node), -1 will be returned.

`lama_input_get_node_unit_name` *unit_id* [*path*] *TEST=name* [Function]
Return a string containing the short name (e.g. "s" for seconds) of the measure unit with ID *unit_id*. The measure unit must belong to the node specified by *path* and to the test named *name*. If *path* is not specified, the current node in the test will be used (see [current_node], page 42). If *name* is not specified, the current test will be used.

Note that this function does not distinguish between X and Y measure units: this means that you can use it either with IDs returned by [lama_input_get_node_x_units], page 41 and [lama_input_get_node_y_units], page 41.

If the node does not contain an unit with ID *unit_id*, an empty string will be returned.

`lama_input_get_node_unit_full_name` *unit_id* [*path*] *TEST=name* [Function]
Return a string containing the full name (e.g. "seconds") of the measure unit with ID *unit_id*. The measure unit must belong to the node specified by *path* and to the test named *name*. If *path* is not specified, the current node in the test will be used (see [current_node], page 42). If *name* is not specified, the current test will be used.

Note that this function does not distinguish between X and Y measure units: this means that you can use it either with IDs returned by [lama_input_get_node_x_units], page 41 and [lama_input_get_node_y_units], page 41.

If the node does not contain an unit with ID *unit_id*, an empty string will be returned.

`lama_input_get_node_unit_description` *unit_id* [*path*] *TEST=name* [Function]

Return a string containing a short description (e.g. "Temperature") of the measure unit with ID *unit_id*. The measure unit must belong to the node specified by *path* and to the test named *name*. If *path* is not specified, the current node in the test will be used (see [current_node], page 42). If *name* is not specified, the current test will be used.

Note that this function does not distinguish between X and Y measure units: this means that you can use it either with IDs returned by [lama_input_get_node_x_units], page 41 and [lama_input_get_node_y_units], page 41.

If the node does not contain an unit with ID *unit_id*, an empty string will be returned.

`lama_input_get_node_type` [*path*] *TEST=name* [Function]

Return the type (an integer) of the node belonging to the test named *name* and specified by the string *path*. If *path* is not specified, the current node in the test will be used (see [current_node], page 42). If *name* is not specified, the current test will be used.

The type codes are listed here:

Code	Description
0	A generic node. This code usually indicates that there is an error in the node tree.
1	The root node of the tree. Its name is the name of the test.
2	A parameter group
3	A node containing photometric calibration information for a channel.
4	A feed horn
5	A radiometric channel
6	A processing type
7	An AUX file
8	A node containing the scientific output of a channel
9	An housekeeping parameter

6.5.3.2 The current node

Each test loaded into memory has a "current working node", much similar to the concept of the "current working directory" in operating systems. This is the default node used by almost all the functions described in this section. The root node is always indicated with `&`. There are two functions designed to set/retrieve the current node: `lama_input_get_current_node` and `lama_input_set_current_node`.

`lama_input_get_current_node` *TEST=name* [Function]
Return the path of the "current" node for the test named *name*. If *name* is not provided, return the current node for the default test.

`lama_input_set_current_node` *path TEST=name* [Function]
Change the current node for the test named *name* to the node specified by *path*. If *name* is not provided, set the current node for the default test.

Return true (nonzero) if the node was found and selected, false (zero) if *path* is not a valid path.

If you want to extract multiple pieces of information from the same node (e.g. the name, the description and the type), it is faster to set that node as the "current" one:

```
;; Print information about the blanking time parameter
x = lama_input_set_current_node ('REBA parameters&LM100326')
print, "Name:        ", lama_input_get_node_name ()
print, "Description: ", lama_input_get_node_description ()
print, "Type:        ", lama_input_get_node_type ()
```

Apart from the small increase in the speed, this also produces cleaner code.

6.5.3.3 Retrieving scientific/housekeeping data

`lama_input_get_hk_node_pair` [*path*] *TEST=name* [Function]
Return the pair of numbers (tab, index) (used by functions like [lama_get_hk_y], page 36) for the housekeeping node specified by *path* and belonging to the test named *name*. If *path* is not specified, use the current node. If *name* is not specified, use the default test.

If the node specified by *path* is not valid, or is not an housekeeping parameter (i.e. the value returned by [lama_input_get_node_type], page 42 is not equal to 9), the function returns -1.

There are two other functions that returns the node path of an AUX/selection/housekeeping files. They are [lama_get_aux_path], page 34, [lama_get_selection_path], page 35 and [lama_get_hk_path], page 37. Refer to sections [IDLScientific], page 28 and [IDLHK], page 36 for further information.

6.5.3.4 Unloading data from memory

Some node types have the ability to load data from FITS files. These are the AUX, scientific and housekeeping nodes (their types are respectively 7, 8 and 9 – see [lama_input_get_node_type], page 42). There is no need for the user to tell when to load the FITS files, because Lama is smart enough to load them when needed.

Starting from version 3.0, Lama gives the user the ability to unload the data loaded from FITS files. This can save a lot of memory, expecially when performing automated tests where a number of radiometric channels and housekeeping parameters must be accessed sequentially. Unloaded data can be reloaded by Lama if they are needed again. Also, if you unload data that are plotted in a window, the window will continue to be usable, as each window maintains a local copy of each data stream.

`lama_input_unload_node` [*path*] *TEST=name* [Procedure]

This function unloads any data associated with the node pointed by *path*. If the node does not contain any datum, the function does nothing.

This function will not free data associated with child nodes – use [lama_input_unload_subtree], page 44.

The following function is a wrapper for [lama_input_unload_node], page 43 which also frees the memory allocated by every descendant of the start node.

`lama_input_unload_subtree` [*path*] *TEST=name* [Procedure]

This function unloads any data associated with the node pointed by *path* and with every descendant. It is implemented in IDL and uses [lama_input_unload_node], page 43.

Consider also using the more specialized functions [lama_input_unload_feed_horn], page 35, [lama_input_unload_aux], page 35, [lama_input_unload_selection], page 35 and [lama_input_unload_hk], page 38.

6.5.3.5 Finding nodes in the tree

Because of the complexity of the node tree, sometimes it is quite difficult to find the desired node. The function `lama_input_find_nodes` provides a powerful way to look into the tree for nodes matching some pattern.

`lama_input_find_nodes` *pattern* [*start_path*] *REGEXP=regexp* [Function]
RECURSIVE=recursive NAME=name TEST=name

This function searches into the list of the children owned by the node specified by *start_path* for any node whose description is equal to *pattern*. If *name* is true, then the node name will be matched instead of the description. If *recursive* is true, all the nodes having *start_path* as ancestor (i.e. its children, grandchildren and so on) will be checked.

The function returns a list of strings, each containing the full path to the found nodes, or an empty string if no matching node was found.

This example will initialize `gain` to the DAE gain states of feed horn 27 channel 00. It finds the right node by looking for an housekeeping parameter matching the `LM` code of the DAE parameter, and then retrieving the tab number and tab index required by [lama_get_hk_y], page 36.

```
;; LM001326 is the code for the DAE gain of FH 27, channel 00
node_path = lama_input_find_nodes ('LM001326', '&', $
                                   /recursive, /name)
pair = lama_input_get_hk_pair (node_path)
gain = lama_get_hk_y (pair[0], pair[1])
```

If *regexp* is nonzero, the pattern will be considered a *regular expression*. This provides an extremely powerful tool to search into a tree. For instance, the following code will print a list of nodes whose description either contains "Feed horn", "feed horn", "FH" or "fh", followed by a two-digit number, with optional space between them:

```
result = lama_input_find_nodes ('([Ff]eed horn|FH|fh)\s*\d\d', $
                                '&', /recursive, /regexp)
```

For more information about regular expressions, see Appendix A [Regular expressions], page 52.

Note that if *regexp* is true, then `lama_input_find_nodes` will return all the nodes whose name matches the regular expression, while if *regexp* is false, it will return all the nodes whose description (or name, if `/name` is used) is *exactly* equal to *pattern*.

Therefore, in this example `r1` and `r2` are not necessarily equal:

```
r1 = lama_input_find_nodes ('DAE gain', '&', /recursive)
r2 = lama_input_find_nodes ('DAE gain', '&', /recursive, /regexp)
```

This happens because if there is a node named `DAE gain for FH27`, it will be included into `r2` but not into `r1`. To make the two searches exactly the same you can use the special characters `^` and `$` (see Appendix A [Regular expressions], page 52 for an explanation of their meaning):

```
;; Now r1 and r2 are the same
r1 = lama_input_find_nodes ('DAE gain', '&', /recursive)
r2 = lama_input_find_nodes ('^DAE gain$', '&', /recursive,
                            /regexp)
```

6.5.4 Using measure units

6.5.4.1 RaNA

RaNA implements a simple way to specify which measure unit to use when asking for scientific data. Functions returning a radiometric output datastream ([lama_get_sky_y], page 29, [lama_get_ref_y], page 29, [lama_selection_get_sky_y], page 31, [lama_selection_get_ref_y], page 31) implement the keywords `RAW` and `PGA_CORRECT` for this purpose. Their meaning is the following:

- If no `PGA_CORRECT` nor `RAW` keyword is specified, the array returned by the function will contain voltages;
- If `PGA_CORRECT` is set to zero, the array returned by the function will contain voltages, but the DAE offset correction will not be applied;
- If `RAW` is specified, the array returned by the function will contain raw ADU values.

The default behavior is therefore to return a stream of voltages.

6.5.4.2 Lama

Lama implements a simple and powerful way to specify measure units when asking for scientific or housekeeping data, by means of the `UNIT` keyword.

The argument for the `UNIT` keyword is a number specifying the measure unit to use.

Here is a list of supported measure units, as well as the function which support them:

`0` Clock ticks ([lama_get_sky_x], page 29, [lama_get_ref_x], page 29, [lama_get_raw_x], page 29, [lama_selection_get_sky_x], page 30, [lama_selection_get_ref_x], page 30, [lama_selection_get_raw_x], page 31, [lama_get_hk_x], page 36)

`1` Seconds [s] ([lama_get_sky_x], page 29, [lama_get_ref_x], page 29, [lama_get_raw_x], page 29, [lama_selection_get_sky_x], page 30,

[lama_selection_get_ref_x], page 30, [lama_selection_get_raw_x], page 31, [lama_get_hk_x], page 36)

`2` Raw radiometer output [ADU] ([lama_get_sky_y], page 29, [lama_get_ref_y], page 29, [lama_get_raw_y], page 29, [lama_selection_get_sky_y], page 31, [lama_selection_get_ref_y], page 31, [lama_selection_get_raw_y], page 31)

`3` Radiometer output [V] ([lama_get_sky_y], page 29, [lama_get_ref_y], page 29, [lama_get_raw_y], page 29, [lama_selection_get_sky_y], page 31, [lama_selection_get_ref_y], page 31, [lama_selection_get_raw_y], page 31)

`4` Radiometer output without DAE offset [V] ([lama_get_sky_y], page 29, [lama_get_ref_y], page 29, [lama_get_raw_y], page 29, [lama_selection_get_sky_y], page 31, [lama_selection_get_ref_y], page 31, [lama_selection_get_raw_y], page 31)

`5` Temperature [K] ([lama_get_hk_y], page 36, when applicable)

`6` Photometric calibration constant [V/K] (this measure unit is for internal use only)

`7` Raw housekeeping data [DEC] ([lama_get_hk_y], page 36, when applicable)

`100` Pure number [] ([lama_get_hk_y], page 36, when applicable)

If you do not specify a `UNIT` keyword, Lama will use an appropriate default choice:

- Time is expressed in seconds (`UNIT=1`);
- Radiometric output is expressed in voltages (`UNIT=3`);
- Housekeeping parameters are expressed using their "natural" measure unit, e.g. temperature (`UNIT=5`).

Please note that Lama supports also the old `RAW` and `PGA_CORRECT` keywords in functions returning radiometric output, but their use is now discouraged in favour of the more general `UNIT` keyword. To upgrade the old code, here are some rules:

1. If no `PGA_CORRECT` nor `RAW` keyword is specified, the result is the same as using `UNIT=3`;
2. If `PGA_CORRECT` is set to zero, the result is the same as using `UNIT=4`;
3. If `RAW` is specified, the result is the same as using `UNIT=2`.

The following function is useful when you get lost while using Lama from the command line. Basically, it provides you the same table shown above.

`lama_dump_measure_units` [Function]

Print a table containing the list of measure units available.

6.6 Triggers

All the trigger functions explained in this section use a peculiar way to specify the LFI channels:

Unit	Code	Unit	Code
FH #27, rad. 0, ch. 0	101	FH #20, rad. 0, ch. 0	129
FH #27, rad. 0, ch. 1	102	FH #20, rad. 0, ch. 1	130

FH #27, rad. 1, ch. 0	103	FH #20, rad. 1, ch. 0	131
FH #27, rad. 1, ch. 1	104	FH #20, rad. 1, ch. 1	132
FH #24, rad. 0, ch. 0	105	FH #19, rad. 0, ch. 0	133
FH #24, rad. 0, ch. 1	106	FH #19, rad. 0, ch. 1	134
FH #24, rad. 1, ch. 0	107	FH #19, rad. 1, ch. 0	135
FH #24, rad. 1, ch. 1	108	FH #19, rad. 1, ch. 1	136
FH #21, rad. 0, ch. 0	109	FH #18, rad. 0, ch. 0	137
FH #21, rad. 0, ch. 1	110	FH #18, rad. 0, ch. 1	138
FH #21, rad. 1, ch. 0	111	FH #18, rad. 1, ch. 0	139
FH #21, rad. 1, ch. 1	112	FH #18, rad. 1, ch. 1	140
FH #22, rad. 0, ch. 0	113	FH #26, rad. 0, ch. 0	141
FH #22, rad. 0, ch. 1	114	FH #26, rad. 0, ch. 1	142
FH #22, rad. 1, ch. 0	115	FH #26, rad. 1, ch. 0	143
FH #22, rad. 1, ch. 1	116	FH #26, rad. 1, ch. 1	144
FH #23, rad. 0, ch. 0	117	Feed horn #18 (70 GHz)	201
FH #23, rad. 0, ch. 1	118	Feed horn #19 (70 GHz)	202
FH #23, rad. 1, ch. 0	119	Feed horn #20 (70 GHz)	203
FH #23, rad. 1, ch. 1	120	Feed horn #21 (70 GHz)	204
FH #25, rad. 0, ch. 0	121	Feed horn #22 (70 GHz)	205
FH #25, rad. 0, ch. 1	122	Feed horn #23 (70 GHz)	206
FH #25, rad. 1, ch. 0	123	Feed horn #24 (44 GHz)	207
FH #25, rad. 1, ch. 1	124	Feed horn #25 (44 GHz)	208
FH #28, rad. 0, ch. 0	125	Feed horn #26 (44 GHz)	209
FH #28, rad. 0, ch. 1	126	Feed horn #27 (30 GHz)	210
FH #28, rad. 1, ch. 0	127	Feed horn #28 (30 GHz)	211
FH #28, rad. 1, ch. 1	128	LFI (all radiometers)	301

This way of numbering channels derives from the FEM channel numeration used by the SCOS2000 scripts to drive the FEM active units (phase switches and LNAs). Since triggers are mainly used for tuning tests, we chose to use the same FEM numeration in the functions dealing with triggers.

`lama_input_get_list_of_triggers` *start end channel time* [Function]

`lama_input_get_trigger` is an high-level procedure wrapped around `lama_input_find_trigger`. It is used to find a list of start/end trigger pairs found in the test. The start trigger is coded by `start`, while the end trigger is coded by `end`. The *time* parameter specifies the start time where to look for triggers.

The return value of this function is a 2 times N matrix where each row contains the start and end time of each trigger. If no triggers with the specified start/end codes exist in the current test, the function returns -1.

```
;; Search for
print, lama_input_get_list_of_triggers (1, 2, 101, 0.0)
```

`lama_input_find_trigger` *channel code time TEST=test* [Function]

`lama_input_find_trigger` looks for a trigger with code *code* and associated with the FEM channel *channel* into the specified test *test* (or the default test, if `TEST` is not specified). The first trigger occurring at a time greater than *time* is returned. If no suitable trigger is found, return -1.

```
;; This example looks for the first start/end trigger in the test
;; and then select data inside. No error checking is made.

start = lama_input_find_trigger (101, 1, 0.0)

;; Note that we start our search from time START instead of 0.0
end_time = lama_input_find_trigger (101, 2, start)

lama_select_range, 0, start, end_time, 1
```

Internally, a trigger is a sequence of values stored in an housekeeping parameter. Setting a trigger in a test is an operation which takes some time (usually a few seconds). The time specified in the *time* parameter always points to the *beginning* of the sequence, while the returned value points to the *end* of the same sequence.

Here is an example. We imagine that in some test the trigger sequence with code 1 was started at 100 s and was made by 6 values, each lasting for 4 seconds. Therefore, the trigger ended at 124 s. Here is how the `lama_input_find_trigger` works in this case:

```
;; Look for trigger 1, starting from 100 s
print, lama_input_find_trigger (101, 1, 100.0)
⊣ 124.0
```

This has several benefits:

1. It is trivial to perform repetitive searches:

```
;; This will print a list of triggers found in the test
;; and referring to the FEM channel 101
time = 0.0
repeat
  time = lama_input_find_trigger (101, 1, time)
  print, "Found a trigger at ", time
until time lt 0.0
```

2. Usually, a test starts after the SCOS operator has finished sending the trigger sequence. Therefore, the value returned by `lama_input_find_trigger` marks exactly the start of the test.

6.7 Log messages

When a test is running, the TQL operator has the ability to write comments that are stored in the test. They are referred as "log messages", and are used to keep track of what the user is doing. Lama provides some IDL functions to access these messages.

`lama_input_get_num_of_log_messages` *TEST=test* [Function]

Return the number of log messages recorded by the TQL for test named *test*. If *test* does not exist, the function returns -1. If there are no log messages recorded, it returns 0.

If the `TEST` keyword was not specified, the default test will be used. See [lama_input_get_selected_test], page 26.

`lama_input_get_log_time` *TEST=test* [Function]

Return an array of double values containing the time (in second) when each log message was recorded by the TQL. These times are comparable with those returned by e.g. [lama_get_sky_x], page 29. The number of elements in the array should match the return value of [lama_input_get_num_of_log_messages], page 48.

If *test* does not exist or if no log message was recorded, the function returns -1.

If the `TEST` keyword was not specified, the default test will be used. See [lama_input_get_selected_test], page 26.

```
;; The first and the last log message usually mark the
;; beginning and the end of the test. We use them to
;; estimate how long the test lasted.
log_time = lama_input_get_log_time ()
num_of_logs = n_elements (log_time)
if num_of_logs gt 0 then begin
    print, "The test lasted", $
           log_time[num_of_logs-1] - log_time[0]
endif
```

`lama_input_get_log_time_str` *TEST=test* [Function]

Return an array of strings containing the date and time when each log message was recorded by the TQL. The number of elements in the array should match the return value of [lama_input_get_num_of_log_messages], page 48. Each string contains the date and time in ISO format, e.g. `yyyy-mm-ddThh:mm:ss` where `yyyy` is the year, `mm` the month, `dd` the day, `hh` the hours, `mm` the minutes and `ss` the seconds when the log message was entered.

If *test* does not exist or if no log message was recorded, the function returns -1.

If the `TEST` keyword was not specified, the default test will be used. See [lama_input_get_selected_test], page 26.

`lama_input_get_log_text` *TEST=test* [Function]

Return an array of strings containing the text of each log message recorded by the TQL. The number of elements in the array should match the return value of [lama_input_get_num_of_log_messages], page 48.

If *test* does not exist or if no log message was recorded, the function returns -1.

If the `TEST` keyword was not specified, the default test will be used. See [lama_input_get_selected_test], page 26.

```
;; Typically the first log message contains a string which
;; describes what the test is about. We want to display it on
;; the screen
if lama_input_get_num_of_log_messages() gt 0 then begin
    print, "Purpose of the test:"
    print, (lama_input_get_log_text())[0]
endif
```

`lama_input_get_log_messages` *TEST=test* [Function]

Return an array of structures containing information about each log message recorded by the TQL. The number of elements in the array should match the return value of [lama_input_get_num_of_log_messages], page 48.

Each element of the array is a structure with three fields:

`time` The same value as returned by [lama_input_get_log_time], page 48.

`time_str` The same value as returned by [lama_input_get_log_time_str], page 49.

`text` The same value as returned by [lama_input_get_log_text], page 49).

If *test* does not exist or if no log message was recorded, the function returns -1.

If the `TEST` keyword was not specified, the default test will be used. See [lama_input_get_selected_test], page 26.

6.8 DAE calibration

The DAE calibration tables are used to convert the digital output produced by the DAE into a voltage. Three tables are required:

1. A '`TSCAL`' file which associates the DAE gain levels with the actual gains.
2. A '`TZERO`' file which includes a (usually small) correction to be applied to the voltage after having applied the '`TSCAL`' table.
3. A '`OFFSET`' file which associates the DAE offset levels with the actual offsets.

Within LAMA, these three tables are named a *calibration set*. Up to LIFE 3.1, only two calibration sets were required: one for the Qualification Model (QM), and the other one for the Flight Model (FM). LAMA decided which calibration table to apply according to the instrument type information read from the test files.

But we must consider that in the future there will be the need to update the calibration tables more than once:

- A new calibration table to be used during the analysis of the tests acquired during the Planck cryogenic campaign must be produced.
- In the CPV phase (the first months after the launch of Planck) we will calibrate the DAE again, therefore producing new tables.

However, we do not want to throw away the old calibration tables, as they will still be needed to load the old tests acquired in the many ground tests performed before the launch. What we want is that for each test recorded during the ground test LAMA applies the "right" calibration tables. Unfortunately, so far the software offered no tools to perform this task.

With version 3.1 we have introduced new facilities that give the user full control over the calibration sets. In LAMA each test can have its own calibration set. Within IDL, each set is identified through a string like `QM`. To get a list of the calibration sets available for a specified test, you can use the `lama_get_calibration_sets` function.

`lama_get_calibration_sets` *TEST=name* [Function]

Return an array of strings representing the list of available calibration sets for the test named *name*. If *path* is not specified, the current test will be used.

If an error occurred during the execution of this function, the value 0 is returned.

The default test can be retrieved and changed with `lama_get_default_calibration` and `lama_default_calibration`.

`lama_get_default_calibration_set` *TEST=name* [Function]

Return a string containing the name of the calibration set being used for the test named *name*. If *path* is not specified, the current test will be used.

If an error occurred during the execution of this function, the value 0 is returned.

`lama_set_default_calibration_set` *new_set TEST=name* [Function]

Select the new calibration set to be used for the test specified by *name* (or for the default test, if the `TEST` keyword is not specified). The value of *new_set* must be one of the strings returned by `lama_get_calibration_sets`.

If an error occurred during the execution of this function, the value 0 is returned.

The following example shows how to use these three functions to set up the calibration set for the default test:

```
;;; Retrieve the list of calibration sets available for the default test
list_of_calibrations = lama_get_calibration_sets ()

;;; Check that the operation was successful (e.g. that at least *one* test
;;; has been loaded...)
if keyword_set (list_of_calibrations) then begin

  ;; We now try to select a new calibration
  if lama_set_default_calibration_set ("CoolestCalibrationEver") then begin

    ;; Done! We have successfully changed the configuration.
    print, "The new DAE calibration is ", lama_get_default_calibration_set ()

  endif else begin

    ;; Something went wrong: warn the user.
    print, "Error: Unable to set the DAE calibration!"

  endelse
endif
```

Of course, in a *real* world you would check that `CoolestCalibrationEver` is one of the supported calibrations by checking that it is included into *list_of_calibrations*.

Appendix A Regular expressions

Regular expressions have a syntax in which a few characters are special constructs and the rest are "ordinary". An ordinary character is a simple regular expression which matches that same character and nothing else. The special characters are `$`, `^`, `.`, `*`, `+`, `?`, `{`, `}`, `[`, `]`, `(`, `)` and `\`. Any other character appearing in a regular expression is ordinary, unless a `\` precedes it.

For example, `f` is not a special character, so it is ordinary, and therefore `f` is a regular expression that matches `f`. Likewise, `o` is a regular expression that matches only `o`.

Any two regular expressions A and B can be concatenated. The result is a regular expression which matches a string if A matches some amount of the beginning of that string and B matches the rest of the string.

As a simple example, we can concatenate the regular expressions `f` and `o` to get the regular expression `fo`, which matches the string `fo`. Still trivial. To do something nontrivial, you need to use one of the special characters. Here is a list of them.

`.` (period) is a special character that matches any single character except a newline. Using concatenation, we can make regular expressions like `a.b`, which matches any three-character string that begins with `a` and ends with `b`.

`*` is not a construct by itself; it is a postfix operator that means to match the preceding regular expression repetitively as many times as possible. Thus, `o*` matches any number of `o` (including no `o`s).

`*` always applies to the *smallest* possible preceding expression. Thus, `fo*` has a repeating `o`, not a repeating `fo`. It matches `f`, `fo`, `foo`, and so on.

The matcher processes a `*` construct by matching, immediately, as many repetitions as can be found. Then it continues with the rest of the pattern. If that fails, backtracking occurs, discarding some of the matches of the `*`-modified construct in case that makes it possible to match the rest of the pattern. For example, in matching `ca*ar` against the string `caaar`, the `a*` first tries to match all three `a`s; but the rest of the pattern is `ar` and there is only `r` left to match, so this try fails. The next alternative is for `a*` to match only two `a`s. With this choice, the rest of the regexp matches successfully.

`+` is a postfix operator, similar to `*` except that it must match the preceding expression at least once. So, for example, `ca+r` matches the strings `car` and `caaaar` but not the string `cr`, whereas `ca*r` matches all three strings.

`?` is a postfix operator, similar to `*` except that it can match the preceding expression either once or not at all. For example, `ca?r` matches `car` or `cr`; nothing else.

`{N}` is a postfix operator that specifies repetition N times–that is, the preceding regular expression must match exactly N times in a row. For example, `x{4}` matches the string `xxxx` and nothing else.

`{N,M}` is a postfix operator that specifies repetition between N and M times–that is, the preceding regular expression must match at least N times, but no more than M times. If M is omitted, then there is no upper limit, but the preceding regular expression must match at least N times.

`( ... | ... )`
: is an *alternation*, that is, it matches any of the regular expression enclosed between the `|`. For instance, `(fh|FH|feed horn)` will match either `fh`, `FH` or `feed horn`. Parentheses is also used in backreferences, see [backreferences], page 54.

`[ ... ]`
: is a *character set*, which begins with `[` and is terminated by `]`. In the simplest case, the characters between the two brackets are what this set can match.

  Thus, `[ad]` matches either one `a` or one `d`, and `[ad]*` matches any string composed of just `a`s and `d`s (including the empty string), from which it follows that `c[ad]*r` matches `cr`, `car`, `cdr`, `caddaar`, etc.

  You can also include character ranges in a character set, by writing the starting and ending characters with a `-` between them. Thus, `[a-z]` matches any lower-case ASCII letter. Ranges may be intermixed freely with individual characters, as in `[a-z$%.]`, which matches any lower-case ASCII letter or `$`, `%` or period.

  Note that the usual regexp special characters are not special inside a character set. A completely different set of special characters exists inside character sets: `]`, `-` and `^`.

  To include a `]` in a character set, you must make it the first character. For example, `[]a]` matches `]` or `a`. To include a `-`, write `-` as the first or last character of the set, or put it after a range. Thus, `[]-]` matches both `]` and `-`.

  To include `^` in a set, put it anywhere but at the beginning of the set. (At the beginning, it complements the set–see below.)

`[^ ... ]`
: `[^` begins a "complemented character set", which matches any character except the ones specified. Thus, `[^a-z0-9A-Z]` matches all characters *except* ASCII letters and digits.

  `^` is not special in a character set unless it is the first character. The character following the `^` is treated as if it were first (in other words, `-` and `]` are not special there).

  A complemented character set can match a newline, unless newline is mentioned as one of the characters not to match. This is in contrast to the handling of regexps in programs such as 'grep'.

`^`
: signifies the beginning of the string. If you wish to match a literal `^` you must escape it by writing `\^`. For example, `^Feed horn` will only match strings which begin with the characters 'Feed horn'.

`$`
: is similar to `^` but matches only at the end of a line. Thus, `x+$` matches a string of one `x` or more at the end of a line.

`\`
: has two functions: it quotes the special characters (including \), and it introduces additional special constructs:

Element	**Meaning**
c	Any character represents itself unless it has a special regexp meaning. Thus c matches the character c.

\c	A character that follows a backslash matches the character itself except where mentioned below. For example if you wished to match a literal caret at the beginning of a string you would write `^`.
\a	This matches the ASCII bell character (BEL, 0x07).
\f	This matches the ASCII form feed character (FF, 0x0C).
\n	This matches the ASCII line feed character (LF, 0x0A, Unix newline).
\r	This matches the ASCII carriage return character (CR, 0x0D).
\t	This matches the ASCII horizontal tab character (HT, 0x09).
\v	This matches the ASCII vertical tab character (VT, 0x0B).
\xhhhh	This matches the Unicode character corresponding to the hexadecimal number hhhh (between 0x0000 and 0xFFFF).
\0ooo (i.e., \zero ooo)	matches the ASCII/Latin1 character corresponding to the octal number ooo (between 0 and 0377).
\d	This matches a digit.
\D	This matches a non-digit.
\s	This matches a whitespace (space or tab).
\S	This matches a non-whitespace.
\w	This matches a word character (a letter, a number or `_`).
\W	This matches a non-word character.
\n	The n-th backreference, e.g. \1, \2, etc.

Because `\` quotes special characters, `\$` is a regular expression that matches only `$`, and `\[` is a regular expression that matches only `[`, and so on.

A.1 Backreferences

Backreferences are used to refer to a previously matched part of the regular expression. When you use constructs like e.g. `(first|1st)` you cannot know which word was matched. By using `\n` (with `n` being a digit) you can backreference a parenthesis. For instance,

`\b(\w+)\W+\1\b` will match `hello--hello` because `\b(\w+)` matches the first `hello`, `\W+` matches `--` and `\1` matches the expression specified by `(\w+)`, which was `hello`.

To use parentheses only to specify alternatives and not for backquoting, you can use the `(?:)`, e.g. `(?:current|voltage)`. This kind of parentheses is slightly more efficient because the parser has less book-keeping to do.

Concept index

A

Add New Test . . . 11
Add to Existing Plot . . . 12
Add to New Plot . . . 11
Analyse Data . . . 9
AUX files . . . 28

B

Binned data . . . 28
Bring to front . . . 13

C

Close Session . . . 9
Close window . . . 13
Combine windows . . . 13
Computing statistics . . . 19
Conventions for this manual . . . 1
Correlation coefficient . . . 22
Cross-correlation plot, Binned data . . . 22
Cross-correlation plot, open . . . 20
Cross-correlation plot, Reset . . . 21
Cross-correlation plot, Save Data . . . 21
Cross-correlation plot, Show error bars . . . 22
Cross-correlation plot, Show fit . . . 22
Cross-correlation plot, Show points . . . 22
Cross-correlation plot, Swap Axes . . . 21
Crosscorrelation plot, open . . . 12

D

DAE calibration, calibration . . . 50

E

Export data to file . . . 12

F

Feed horns . . . 27
Full scientific data . . . 28

H

help . . . 3

I

IDL functions, Feed horns . . . 27

L

Lama, example . . . 3
Lama, open . . . 9
Load Data . . . 12, 16

M

Make Default . . . 12
Manual Zoom . . . 19
Measure units . . . 45
Messages, Log . . . 48

N

Node List . . . 10
Node paths . . . 39
Normalize data . . . 15

O

Open Object Information . . . 12
Overview . . . 1

P

Path to nodes . . . 39

R

RaNA . . . 8
Regular expressions . . . 44, 52
Remove Mean . . . 15
Report . . . 9

S

Scientific data . . . 28
Show Data . . . 16
Show Gaps . . . 18
Start Session . . . 9
Swap sky/ref . . . 18

T

Triggers . . . 46
Tutorial . . . 3

U

Units of measure . . . 45
Unload Data . . . 12, 16
Unload Test . . . 11

V

View Data . . . 9

Function index

lama_channel_is_swapped 28
lama_dump_hk_tables 37
lama_dump_measure_units 46
lama_get_aux_path 35
lama_get_calibration_sets 50
lama_get_default_calibration_set 51
lama_get_env_ave 37
lama_get_env_sig 37
lama_get_hk_description 36
lama_get_hk_name 36
lama_get_hk_path 37
lama_get_hk_x 36
lama_get_hk_y 37
lama_get_num_of_sci_tabs 28
lama_get_raw_sigma 30
lama_get_raw_x 29
lama_get_raw_y 30
lama_get_ref_sigma 29
lama_get_ref_x 29
lama_get_ref_y 29
lama_get_sci_ave 32
lama_get_sci_nsamples 32
lama_get_sci_sig 32
lama_get_selection_path 35
lama_get_sky_sigma 29
lama_get_sky_x 29
lama_get_sky_y 29
lama_get_statistic_range 33
lama_input_find_nodes 44
lama_input_find_trigger 47
lama_input_get_children 40
lama_input_get_current_node 43
lama_input_get_hk_node_pair 43
lama_input_get_list_of_tests 27
lama_input_get_list_of_triggers 47
lama_input_get_log_messages 50
lama_input_get_log_text 49
lama_input_get_log_time 49
lama_input_get_log_time_str 49
lama_input_get_node_description 41
lama_input_get_node_name 40
lama_input_get_node_type 42
lama_input_get_node_unit_description 42
lama_input_get_node_unit_full_name 41
lama_input_get_node_unit_name 41
lama_input_get_node_x_units 41
lama_input_get_node_y_units 41
lama_input_get_num_of_log_messages 48
lama_input_get_selected_feed_horn 27
lama_input_get_selected_test 26
lama_input_is_test_available 26
lama_input_load_test 26
lama_input_select_feed_horn 27
lama_input_select_test 26
lama_input_set_current_node 43
lama_input_unload_aux 35
lama_input_unload_feed_horn 35
lama_input_unload_hk 38
lama_input_unload_node 44
lama_input_unload_selection 36
lama_input_unload_subtree 44
lama_input_unload_test 26
lama_instance_close 25
lama_instance_create 25
lama_instance_is_created 25
lama_is_channel_calibrated 34
lama_is_statistic_range_selected 33
lama_select_range 30
lama_selection_get_binning 33
lama_selection_get_nsamples 33
lama_selection_get_range 33
lama_selection_get_raw_sigma 31
lama_selection_get_raw_x 31
lama_selection_get_raw_y 31
lama_selection_get_ref_sigma 31
lama_selection_get_ref_x 31
lama_selection_get_ref_y 31
lama_selection_get_sci_avg 33
lama_selection_get_sci_sigma 33
lama_selection_get_sky_sigma 31
lama_selection_get_sky_x 30
lama_selection_get_sky_y 31
lama_selection_is_ref_available 30
lama_selection_is_sky_available 30
lama_set_calibration 34
lama_set_default_calibration_set 51
lama_swap_sky_ref 28
lama_view_close 26
lama_view_is_opened 26
lama_view_open 25